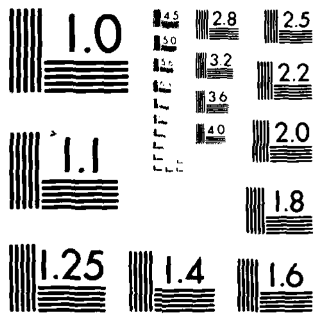


MICROCOPY RESOLUTION TEST CHART

September 1981 Report. No. STAN-CS-81-843

2

Verification of Concurrent Programs, Part II: Temporal Proof Principles

by

Zohar Manna
Amir Pnueli

Department of Computer Science

Stanford University
Stanford, CA 94305

LELAND STANFORD JUNIOR UNIVERSITY
ORGANIZED 1891

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR-TR- 82-0495	2. GOVT ACCESSION NO. AD-A116 035	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) VERIFICATION OF CONCURRENT PROGRAMS, PART II: TEMPORAL PROOF PRINCIPLES		5. TYPE OF REPORT & PERIOD COVERED TECHNICAL
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Z. Manna A. Pnueli		8. CONTRACT OR GRANT NUMBER(s) AFOSR-81-0014
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Computer Science Artificial Intelligence Laboratory Stanford University, Stanford, CA 94305		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61102F 2304/A2
11. CONTROLLING OFFICE NAME AND ADDRESS Mathematical & Information Sciences Directorate Air Force Office of Scientific Research Bolling AFB, DC 20332		12. REPORT DATE SEP 81
		13. NUMBER OF PAGES 51
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Appears in the Proceedings of the Workshop on Logics of Programs, Yorktown Heights, NY, Springer-Verlag Lecture Notes in Computer Science, 1981

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

In this paper, the second of a series on the application of temporal logic to concurrent programs, we present proof methods for establishing invariance (safety) and eventuality (liveness) properties.
The proof principle for establishing invariance properties is based on computational induction, and is a generalization of the inductive assertion method. For a restricted class of concurrent programs we present an algorithm for the automatic derivation of invariant assertions. (over)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. (cont.)

In order to establish eventuality properties we present several proof principles that translate the structure of the program into basic temporal statements about its behavior. These principles can be viewed as providing the temporal semantics of the program. The basic statements thus derived are then combined into temporal proofs for the establishment of eventuality properties. This method generalizes the intermittent assertion method.

The proof principles are amply illustrated by examples.

VERIFICATION OF CONCURRENT PROGRAMS: TEMPORAL PROOF PRINCIPLES

by

ZOHAR MANNA
Computer Science Department
Stanford University
Stanford, CA
and
Applied Mathematics Department
The Weizmann Institute
Rehovot, Israel

AMIR PNUELI
Applied Mathematics Department
The Weizmann Institute
Rehovot, Israel

Abstract

In this paper, the second of a series on the application of temporal logic to concurrent programs, we present proof methods for establishing *invariance* (*safety*) and *eventuality* (*liveness*) properties.

The proof principle for establishing invariance properties is based on computational induction, and is a generalization of the ***inductive assertion*** method. For a restricted class of concurrent programs we present an algorithm for the automatic derivation of invariant assertions.

In order to establish eventuality properties we present several proof principles that translate the structure of the program into basic temporal statements about its behavior. These principles can be viewed as providing the temporal semantics of the program. The basic statements thus derived are then combined into temporal proofs for the establishment of eventuality properties. This method generalizes the ***intermittent assertion*** method.

The proof principles are amply illustrated by examples.

The first paper in this series, the ***temporal framework*** part, appears in *The Correctness Problem in Computer Science* (R. S. Boyer and J S. Moore, eds.), International Lecture Series in Computer Science, Academic Press, London, 1981.

This paper appears in the ***Proceedings of the Workshop on Logics of Programs (Yorktown-Heights, NY)***, Springer-Verlag Lecture Notes in Computer Science, 1981.

This research was supported in part by the National Science Foundation under grants MCS79-09495 and MCS80-06930, by the Office of Naval Research under Contract N00014-76-C-0687, and by the United States Air Force Office of Scientific Research under Grant AFOSR-81-0014.

INTRODUCTION

In a previous report [MP2] we introduced the temporal framework for reasoning about concurrent programs. We described the model of concurrent programs that we study which is based on interaction via shared variables and defined the concept of fair execution of such programs. We then demonstrated the application of the temporal logic formalism to the *expression* of properties of concurrent programs. Program properties of interest can be classified according to the syntactic form of the temporal formula expressing them; we studied three classes of properties: invariance properties, eventuality properties and precedence properties. We have shown that almost all of the program properties that were ever considered or studied for either sequential or concurrent programs fall into one of these three categories. These include properties such as partial correctness, clean behavior, global invariants, mutual exclusion, safety, deadlock absence, output integrity – in the invariance category; total correctness, intermittent assertion realization, accessibility, liveness, responsiveness – in the eventualities category; and safe liveness, absence of unsolicited response, FIFO responsiveness and general precedence – in the precedence category.

In this paper, a sequel to [MP2], we concentrate on the application of the temporal logic formalism to *proving* these properties. We would thus present methods for establishing that a given program indeed possesses a certain property. In principle, once a property has been expressed within the temporal logic formalism, and an appropriate temporal characterization of the behavior of the given program derived ([MAN1], [MP1], [PNU1], [PNU2]), the task of proving that the property holds for this program reduces to proving the validity of a certain temporal implication. This implication states that every sequence of states, if it is a fair computation of the given program, has the desired property.

These principles can be justified by the general temporal formalism, and once justified, provide direct, simple, and intuitive rules for the establishment of these properties. They usually replace long but repetitively similar chains of primitive steps in more detailed proofs, and help us focus on the higher level overview of the proof while retaining the necessary standard of rigor.

Previous attempts to develop proof techniques for concurrent programs include [KEL], [LAM] and [OG].

In our exposition, we assume that the reader is familiar with the concepts and definitions introduced in our first paper of this series – [MP2].

THE INVARIANCE PRINCIPLE

Consider a typical concurrent program P of form

$$(\bar{y} := f_0(\bar{x}));\ [P_1 \| \cdots \| P_m]$$

with input parameters $\bar{x} = (x_1, \ldots, x_k)$ and shared program variables $\bar{y} = (y_1, \ldots, y_n)$ over a domain D. Let ψ be a classical formula, *i.e.*, a formula with no modal operators. The basic idea in proving that the formula ψ is an invariant of the program P, *i.e.*

$$\models\ \varphi(\bar{x}) \supset \Box\psi,$$

is to show that:

(*a*) the precondition $\varphi(\overline{x})$ implies that ψ is true initially.

(*b*) ψ is preserved by any possible transition of the program P; that is, if it were true before the transition then it also will be true after the transition.

We can then infer the invariance of ψ under the precondition $\varphi(\overline{x})$.

To state the result more precisely, let $Q(\overline{\pi};\overline{y})$ be a "state property", *i.e.*, it is expressed by a classical formula with no temporal operators, which may refer to the location variables $\overline{\pi}$, the program variables $\overline{y}$, and possibly some global variables.

Let

$$\begin{tikzcd} \ell \arrow[rr, "c_\alpha(\overline{y}) \to {[\overline{y} := f_\alpha(\overline{y})]}", "\alpha"'] & & \ell' \end{tikzcd}$$

be a transition in process P_j for some $j = 1, \ldots, m$. With each such transition we associate the location transformation function r_α given by:

$$r_\alpha(\pi_1, \ldots, \pi_j, \ldots, \pi_m) = (\pi_1, \ldots, \ell', \ldots, \pi_m),$$

i.e., the value of π_j is replaced by ℓ', while the value of each π_i, $i \neq j$, is unchanged. This transformation denotes the change in the vector $\overline{\pi}$ when transition α is taken, much in the same way that f_α denotes the change in $\overline{y}$ when α is taken.

The notation we use to express the location change as a transformation underlines the similarity between the location and program variables. This leads to the possible description of a transition as:

$$\begin{tikzcd} \bigcirc \arrow[rr, "{[at\,\ell \wedge c_\alpha(\overline{y})] \to [(\overline{\pi};\overline{y}) := (r_\alpha(\overline{\pi}); f_\alpha(\overline{y}))]}", "\alpha"'] & & \bigcirc \end{tikzcd}$$

A property $Q(\overline{\pi};\overline{y})$ is said to be ***inductive*** for P if the following ***verification condition*** holds for each transition α in P:

$$V_\alpha : \quad [at\,\ell \wedge c_\alpha(\overline{y}) \wedge Q(\overline{\pi};\overline{y})] \quad \supset \quad Q\big(r_\alpha(\overline{\pi}); f_\alpha(\overline{y})\big).$$

Intuitively, Q is inductive if it is inherited along every transition *i.e.*, if it was true before the transition and the transition was enabled, it will necessarily be true after the transition. Note that the verification condition is classical, in the sense that it contains no temporal operators, and can therefore be established using classical proof techniques.

Our proof rule for invariance may now be formulated as follows:

The Invariance Principle

Let $Q(\overline{\pi};\overline{y})$ be a state property of a program P such that:

1. Q is true initially; *i.e.*,

$$I:\quad [at\,\overline{\ell_0} \land \varphi(\overline{x})] \supset Q(\overline{\pi}; f_0(\overline{x}))$$

holds, where $\overline{\ell_0} = (\ell_0^1, \ldots, \ell_0^m)$ the vector of initial locations.

2. Q is inductive for P; *i.e.*, the verification condition

$$V_\alpha:\quad [at\,\ell \land c_\alpha(\overline{y}) \land Q(\overline{\pi};\overline{y})] \supset Q(r_\alpha(\overline{\pi}); f_\alpha(\overline{y}))$$

holds for every transition α in P.

Then we may deduce

$$\models\ [at\,\overline{\ell_0} \land \varphi(\overline{x})] \supset \square Q(\overline{\pi};\overline{y}).$$

Condition 1 ensures that Q is true initially, provided we restrict ourselves to inputs $\overline{x}$ satisfying φ and condition 2 ensures that once Q is true it remains so. The conclusion is that Q is invariantly true for all (P, φ)-computations.

Note that this proof principle reduces the proof of a temporal formula of the invariance class into a classical proof of a set of formulas, namely the initial condition I and the verification conditions V_α.

The principle of invariance described here is the most general method known for proving invariance properties of concurrent programs. It can be shown to underlie all other proposed proof methods for invariance properties.

PRAGMATIC CONSIDERATIONS IN CHECKING FOR INDUCTIVENESS

In principle, when checking for the inductiveness of an assertion Q one has to check the verification condition V_α for all transitions α in the program. However, in practice, we can immediately discard many transitions as automatically preserving Q, based on syntactic considerations alone.

If the property Q does not contain any of the location variables $\overline{\pi}$, then the required verification conditions V_α are reduced to

$$V'_\alpha:\quad [c_\alpha(\overline{y}) \land Q(\overline{y})] \supset Q(f_\alpha(\overline{y})).$$

In particular, V'_α is trivially true for any transition α where f_α does not modify the variables on which Q actually depends.

A typical case is that of semaphores. We have the following property:

The Semaphore Variable Rule: For a semaphore variable y,

if its initial value is a nonnegative integer
and if it is modified only by ***request*** and ***release*** instructions,
then

$$\models \ \square(y \geq 0).$$

The only two instructions that may modify the value of a semaphore variable are:

request(y), which is equivalent to

$$\begin{tikzcd} \bigcirc \arrow[r, "y > 0 \rightarrow [y := y - 1]"] & \bigcirc \end{tikzcd}$$

and *release*(y), which is equivalent to

$$\begin{tikzcd} \bigcirc \arrow[r, "\textit{true} \rightarrow [y := y + 1]"] & \bigcirc \end{tikzcd}$$

For the ***request*** case the verification condition is

$$[(y > 0) \ \wedge \ (y \geq 0)] \ \supset \ (y - 1 \geq 0).$$

For the ***release*** transition the verification condition is

$$[\textit{true} \ \wedge \ (y \geq 0)] \ \supset \ (y + 1 \geq 0).$$

Both conditions are trivially true. Thus, since the initial value of the semaphore variable y is nonnegative and it is modified only through the semaphore instructions *request*(y) and *release*(y), it follows, by the Invariance Principle, that y is invariantly nonnegative, *i.e.* $\models \square(y \geq 0)$.

For another example, let us consider a typical assertion of the form:

$$Q(\bar{\pi}; \bar{y}) : \quad at\,L \ \supset \ \phi(\bar{y}),$$

where L is a set of locations in P and ϕ does not depend on the location variables. For an arbitrary transition α of the form

$$\begin{tikzcd} \ell \arrow[r, "c_\alpha(\bar{y}) \ \rightarrow \ [\bar{y} := f_\alpha(\bar{y})]", "\alpha"'] & \ell' \end{tikzcd}$$

the verification condition is

$$V_\alpha : \quad \{c_\alpha(\bar{y}) \ \wedge \ [(\ell \in L) \ \supset \ \phi(\bar{y})]\} \ \supset \ [(\ell' \in L) \ \supset \ \phi(f_\alpha(\bar{y}))],$$

or equivalently,

$$\{c_\alpha(\bar{y}) \ \wedge \ [(\ell \not\in L) \ \vee \ \phi(\bar{y})] \ \wedge \ (\ell' \in L)\} \ \supset \ \phi(f_\alpha(\bar{y})).$$

There are three cases to consider.

Case: $\ell' \notin L$ (outside or leaving L). Then V_α is trivially true, since the antecedent of the implication is false.

Case: $\ell \notin L$, $\ell' \in L$ (entering L). Then V_α is reduced to

$$c_\alpha(\bar{y}) \supset \phi(f_\alpha(\bar{y})).$$

Case: $\ell, \ell' \in L$ (within L). Then V_α is reduced to

$$[c_\alpha(\bar{y}) \wedge \phi(\bar{y})] \supset \phi(f_\alpha(\bar{y})).$$

Thus, we only have to consider α's which fall into the two latter cases.

EXAMPLE: CONSUMER-PRODUCER

Let us illustrate an application of the invariance principle to the Consumer-Producer program (program *CP* of [MP2]).

$$b := \Lambda, \quad s := 1, \quad cf := 0, \quad ce := N$$

ℓ_0 :	*compute* y_1	m_0 :	*request*(cf)
ℓ_1 :	*request*(ce)	m_1 :	*request*(s)
ℓ_2 :	*request*(s)	m_2 :	$y_2 := head(b)$
ℓ_3 :	$t_1 := b \circ y_1$	m_3 :	$t_2 := tail(b)$
ℓ_4 :	$b := t_1$	m_4 :	$b := t_2$
ℓ_5 :	*release*(s)	m_5 :	*release*(s)
ℓ_6 :	*release*(cf)	m_6 :	*release*(ce)
ℓ_7 :	*go to* ℓ_0	m_7 :	*compute using* y_2
		m_8 :	*go to* m_0
	— P_1 : Producer —		— P_2 : Consumer —

The producer P_1 computes a value into y_1 without using any other program variables; the computation details being irrelevant. It then adds y_1 to the end of the buffer b. The consumer P_2 removes the first element of the buffer into y_2 and then uses this value for its own purposes (at m_7). It is assumed that the maximal capacity of the buffer b is $N > 0$. The '*compute using* y_2' instruction references y_2 but does not modify any of the shared program variables.

In order to ensure the correct synchronization between the processes we use three semaphore variables: The variable s ensures that the accesses to the buffer are protected and provides exclusion between the sections (ℓ_3, ℓ_4, ℓ_5) and (m_2, m_3, m_4, m_5). The variable ce ("count of empties") counts the number of free available slots in the buffer b. The variable cf ("count of fulls") counts how many items the buffer currently holds.

The initial condition is given by:

$$at\,\ell_0 \;\wedge\; at\,m_0 \;\wedge\; (b = \Lambda) \;\wedge\; (s = 1) \;\wedge\; (cf = 0) \;\wedge\; (ce = N).$$

We will use invariances to prove several properties of this program.

First, we observe that due to the semaphore variable rule

(1) $\models \Box[(s \geq 0) \wedge (cf \geq 0) \wedge (ce \geq 0)]$.

Mutual Exclusion

The exclusive access to the critical sections

$$L = \{\ell_3, \ell_4, \ell_5\}$$
$$M = \{m_2, m_3, m_4, m_5\}$$

can be expressed as:

$$\models \Box \sim(at\,L \wedge at\,M),$$

i.e., it is never the case that $\pi_1 \in L$ and $\pi_2 \in M$ simultaneously.

Since only one $at\,\ell_i$ and only one $at\,m_i$ can be true at a given instant it is sufficient to prove:

(2) $\models \Box[(at\,L + at\,M) \leq 1]$.

Note the mixed notation that treats propositions as numerically valued with *true* = 1, *false* = 0.

Formula (2) states an invariance property. It will be proved by showing the invariance of the assertion:

$$Q_1 : \qquad at\,L + at\,M + s = 1.$$

By the invariance principle we have to show that Q_1 is true initially and that Q_1 is inductive for P.

Initially, we have that $s = 1$ and that $at\,\ell_0 = at\,m_0 = 1$ which implies that $at\,L = at\,M = 0$. Thus the left-hand side of the equality in Q_1 evaluates to 1 and we have that Q_1 holds initially.

Next, we have to check that Q_1 is inductive, *i.e.*, preserved by every transition in P. From inspection of the variables on which Q_1 depends, it is clear that it is sufficient to check the transitions that either modify s or modify the $at\,L$ or $at\,M$ propositions. The only candidates for modifying Q_1 are therefore the transitions $\ell_2 \rightarrow \ell_3$, $\ell_5 \rightarrow \ell_6$, $m_1 \rightarrow m_2$, and $m_5 \rightarrow m_6$.

Take, for example, the transition $\ell_2 \rightarrow \ell_3$. Going through this transition changes $at\,L$ from 0 to 1 increasing the sum by 1. But, as s is decremented by 1, the sum remains constant. Similar checks of the other transitions will show that they all leave the sum invariant. This establishes the inductiveness of Q_1.

We may therefore conclude by the Invariance Principle that

$$\models \ \Box Q_1$$

i.e., Q_1 is an invariant of the program P.

The combination of $\Box Q_1$ and the semaphore property $\Box(s \geq 0)$ implies property (2) that proves mutual exclusion.

Proper Management of the Buffer

Here we would like to show that

(3) $$\models \ \Box(0 \leq |b| \leq N),$$

i.e., the buffer's maximum capacity is never exceeded throughout the execution and no attempt is made to remove an element from an empty buffer.

We first establish the invariance of the following inductive assertion:

$$Q_2: \quad cf + ce + at\,\ell_{2..6} + at\,m_{1..6} \ = \ N$$

We use here our abbreviated notation, where $at\,\ell_{2..6}$ stands for $at\{\ell_2, \ldots, \ell_6\}$, *i.e.*, $\pi_1 \in \{\ell_2, \ldots, \ell_6\}$, and $at\,m_{1..6}$ stands for $at\{m_1, \ldots, m_6\}$, *i.e.*, $\pi_2 \in \{m_1, \ldots, m_6\}$. As before, the whole conjunction is interpreted arithmetically: 1 standing for *true* and 0 for *false*. By inspection of the relevant transitions we verify that Q_2 is indeed inductive and initially true, and thus is invariant, *i.e.*,

$$\models \ \Box Q_2.$$

Next consider another necessary invariant assertion:

$$Q_3: \quad cf + at\,\ell_{5,6} + at\,m_{1..4} \ = \ |b|,$$

where $|b|$ is the size of the buffer b. To establish the invariance of Q_3 we have to also establish the invariance of

$$Q_4: \quad at\,\ell_4 \ \supset \ \big(|t_1| = |b| + 1\big)$$

and

$$Q_5: \quad at\,m_4 \ \supset \ \big(|t_2| + 1 = |b|\big).$$

We will check for the joint invariance of Q_3, Q_4, and Q_5 and establish $\models \Box(Q_3 \wedge Q_4 \wedge Q_5)$.

The conjunction $Q_3 \wedge Q_4 \wedge Q_5$ is initially of the form $(0 = 0) \wedge (\mathit{false} \supset \ldots) \wedge (\mathit{false} \supset \ldots)$ which is clearly true.

In order to check the inductiveness of $Q_3 \wedge Q_4 \wedge Q_5$ we must check every relevant transition of the program CP. Let us consider two typical transitions:

$\ell_3 \to \ell_4$:

Q_3 and Q_5 are not affected at all. In Q_4, both $at\,\ell_4$ and $|t_1| = |b| + 1$ become true on this transition, so that Q_4 is true after the transition.

$\ell_4 \to \ell_5$:

Here, Q_3, Q_4, and Q_5 are all affected by the transition and we would like, therefore, to illustrate the proof of a verification condition along this transition in greater detail. The verification condition is:

$$[\, at\,\ell_4 \;\wedge\; Q_3(\overline{\pi};\overline{y}) \;\wedge\; Q_4(\overline{\pi};\overline{y}) \;\wedge\; Q_5(\overline{\pi};\overline{y})\,]$$
$$\supset \;[\, Q_3(r(\overline{\pi}); f(\overline{y})) \wedge Q_4(r(\overline{\pi}); f(\overline{y})) \wedge Q_5(r(\overline{\pi}); f(\overline{y}))\,]$$

where

$$r(\pi_1, \pi_2) = (\ell_5, \pi_2)$$
$$f(b, s, cf, ce, t_1, t_2) = (t_1, s, cf, ce, t_1, t_2).$$

The proof proceeds in the following steps:

1.	$at\,\ell_4$	given
2.	$at\,\ell_{5,6} = 0$	from 1
3.	$cf + at\,m_{1..4} = \|b\|$	by Q_3
4.	$\|t_1\| = \|b\| + 1$	by Q_4 using 1
5.	$cf + 1 + at\,m_{1..4} = \|b\| + 1$	by adding 1 to both sides of 3
6.	$cf + (\ell_5 \in \{\ell_5, \ell_6\}) + at\,m_{1..4} = \|t_1\|$	from 5 using 4
7.	$Q_3(r(\overline{\pi}); f(\overline{y}))$	by definition of r and f using Q_3

Consider next $Q_4(r(\overline{\pi}); f(\overline{y}))$:

8.	$(\ell_5 = \ell_4) \supset (\|t_1\| = \|t_1\| + 1)$	tautology
9.	$Q_4(r(\overline{\pi}); f(\overline{y}))$	by definition of r and f using Q_4

As for $Q_5(r(\overline{\pi}); f(\overline{y}))$:

10.	$\sim at\,m_4$	by 1 and mutual exclusion (2)
11.	$at\,m_4 \supset (\|t_2\| + 1 = \|t_1\|)$	from 12
12.	$Q_5(r(\overline{\pi}); f(\overline{y}))$	by definition of r and f using Q_5

This concludes the proof of the verification condition for transition $\ell_4 \rightarrow \ell_5$. Therefore $Q_3 \wedge Q_4 \wedge Q_5$ is inductive along the transition $\ell_4 \rightarrow \ell_5$. We can similarly check that it is inductive along all the other transitions.

Thus we have established:

$$\models \quad \Box(Q_3 \wedge Q_4 \wedge Q_5).$$

Let us now proceed to infer the proper management of the buffer b, *i.e.*, $\Box(0 \leq |b| \leq N)$.

First observe that by Q_3, $|b|$ is equal to a sum of variables all of which are nonnegative. Thus we have

$$\models \quad \Box(|b| \geq 0).$$

On the other hand we have by Q_3 and Q_2 that

$$\begin{aligned} & |b| - cf \\ = \quad & at\ell_{5,6} + atm_{1..4} \\ \leq \quad & at\ell_{2..6} + atm_{1..6} \\ = \quad & N - (cf + ce) \end{aligned}$$

The first equality is a direct consequent of Q_3. The inequality results from the fact that $\{\ell_5, \ell_6\}$ is a subset of $\{\ell_2, \ldots, \ell_6\}$ and $\{m_1, \ldots, m_4\}$ is a subset of $\{m_1, \ldots, m_6\}$. The second equality is a direct consequence of Q_2.

Thus, we have

$$|b| - cf \leq N - (cf + ce)$$

which simplifies to

$$|b| \leq N - ce.$$

Since ce is a semaphore variable we have $ce \geq 0$ which gives

$$\models \quad \Box(|b| \leq N).$$

Thus we conclude that property (3),

$$\models \quad \Box(0 \leq |b| \leq N),$$

holds.

Comments

• ***Modifying the program***

The need for the auxiliary invariants Q_4 and Q_5 resulted from the splitting of the statements concerning b into several statements according to the single-access rule.

Having first established the mutual exclusion of the regions $L = \{\ell_3, \ell_4, \ell_5\}$ and $M = \{m_2, \ldots, m_5\}$ we can observe that b is not really a shared variable, in that only one process at a time can access it. Correspondingly, we could transform the program, after having established exclusion, by replacing

$$\begin{array}{ll} \ell_3 : & t_1 := b \circ y_1 \\ \ell_4 : & b := t_1 \end{array}$$

by

$$\ell_3' : \; b := b \circ y_1$$

and

$$\begin{array}{ll} m_2 : & y_2 := head(b) \\ m_3 : & t_2 := tail(b) \\ m_4 : & b := t_2 \end{array}$$

by

$$m_2' : \; (y_2, b) := \big(head(b),\ tail(b)\big).$$

This would greatly simplify the subsequent analysis by making Q_3 directly verifiable without using Q_4 and Q_5.

• ***Using virtual variables***

Instead of introducing the auxiliary invariants Q_4, Q_5 it is possible to define a virtual variable b^* by:

$$b^* \quad = \quad \textit{if} \;\; at\,\ell_4 \;\; \textit{then} \;\; t_1 \;\; \textit{else} \;\; (\textit{if} \;\; at\,m_4 \;\; \textit{then} \;\; t_2 \;\; \textit{else} \;\; b)$$

and then directly prove a modified version of Q_3:

$$Q_3^* : \quad cf + at\,\ell_{4..6} + at\,m_{1..3} \;=\; |b^*|.$$

The variable b^* represents the intended value of b, where we use t_i $(i = 1, 2)$ instead of b if b is about to be changed to t_i. Because we are focusing our attention on the value as soon as it is obtained, we have modified Q_3 by extending the region $\{\ell_5, \ell_6\}$ into $\{\ell_4, \ell_5, \ell_6\}$ and contracting $\{m_1, m_2, m_3, m_4\}$ into $\{m_1, m_2, m_3\}$.

A SYSTEMATIC SEARCH FOR LINEAR INVARIANTS

In order to dispel the illusion of "magically" drawing the invariants Q_1, Q_2, Q_3 out of thin air, let us describe a method for a systematic search for such invariants. (See also [FRA], [CLA].)

An invariant of the form discussed here is composed of three parts, such that the sum of the first two is equal to the third. We represent such an invariant by:

$$(B + Z) = C.$$

(*a*) B is the *body* of the invariant and is a linear expression in the semaphore variables and other variables which are incremented by constants (linearly) during cycles in the program.

(*b*) Z is a sum of expressions of the form $\pi_j \in L$ for some region $L \subseteq \mathcal{L}_j$ and will be called a ***compensation expression.***

(*c*) C is a constant.

We start constructing such an invariant by finding an appropriate body.

(*a*) In the body we look for a linear combination of variables $E = \sum a_i y_i$ such that the net change in each cycle of each process is 0. Obviously, we restrict ourselves to cyclic programs, *i.e.*, non-terminating programs, in which each process eventually returns to its initial location ℓ_0 and to variables whose change along a cycle is constant and independent of the program flow. Semaphore variables usually have this property.

Let us denote for these variables the net change in y_i resulting from a full cycle in process P_j by Δ_i^j. Then our combination $E = \sum a_i y_i$ should satisfy

$$\Delta^j E = \Sigma a_i \Delta_i^j = 0$$

for j, $0 \leq j \leq m$. That is, we require that the value of the expression remains unchanged as a result of a complete cycle of each of the processes.

In our consumer-producer example all our variables are linearly incremented and we have the following table:

$\Delta_s^1 = 0$	$\Delta_s^2 = 0$
$\Delta_{\lvert b \rvert}^1 = 1$	$\Delta_{\lvert b \rvert}^2 = -1$
$\Delta_{cf}^1 = 1$	$\Delta_{cf}^2 = -1$
$\Delta_{ce}^1 = -1$	$\Delta_{ce}^2 = 1.$

We look for a combination

$$E = a_1 \cdot s + a_2 \cdot |b| + a_3 \cdot cf + a_4 \cdot ce$$

such that $\sum a_i \Delta_i^j = 0$ for $j = 1, 2$. This yields the set of equations

$$\begin{aligned} a_1 \cdot 0 + a_2 + a_3 - a_4 &= 0 \\ a_1 \cdot 0 - a_2 - a_3 + a_4 &= 0. \end{aligned}$$

We will be interested in a nontrivial set of independent solutions to these equations.

In this case the equations possess three degrees of freedom, and hence three linearly independent solutions are possible. The exact choice is irrelevant and we pick the following:

1. $a_1 = 1 \qquad a_2 = a_3 = a_4 = 0$
2. $a_3 = a_4 = 1 \qquad a_1 = a_2 = 0$
3. $a_2 = a_4 = 1 \qquad a_1 = a_3 = 0.$

Thus for the following independent linear combinations, the net change in each cycle of each process is 0:

$$B_1 : \; s$$
$$B_2 : \; cf + ce$$
$$B_3 : \; |b| + ce.$$

Note that B_1 and B_2 correspond to the bodies of Q_1 and Q_2 respectively, while B_3 is a different invariant which will enable us to derive the same conclusion as the combination of Q_2 and Q_3. For the choice $a_1 = a_4 = 0$, $a_2 = -1$ and $a_3 = 1$, we could get $B'_3 : cf - |b|$ which corresponds to Q_3 itself.

(*b*) Having a body B, to derive the right-hand side C of the invariant, we only have to substitute the initial values implied by $\varphi(\overline{x})$ into the body. Doing this for our three invariants we obtain:

$$C_1 : \; 1$$
$$C_2 : \; N$$
$$C_3 : \; N.$$

(*c*) Next, we determine the compensation expressions. Consider a given C and $B(\overline{y})$ and a process P_j with locations $\{\ell_0, \ldots, \ell_e\}$. Since we assume cycling, ℓ_e is not a terminal location but branches back to ℓ_0. By our assumption, the changes in $B(\overline{y})$ can be traced and are constant. Denote by $B_i(\overline{y})$ the value of B at location ℓ_i, $i = 0, 1, \ldots e$ in the process, and let

$$\delta_i \;=\; B_0(\overline{y}) - B_i(\overline{y}).$$

Then the compensating expression for process P_j is given by

$$Z_j \;=\; \sum_{i=0}^{e} \delta_i \cdot (at\,\ell_i).$$

For example, to evaluate δ_i for $B_1 = s$ in P_1 above we have to compute:

$$s|_{at\,\ell_0} - s|_{at\,\ell_i}.$$

Assuming that P_1 is operating alone, (which is the basic assumption in the computation of the δ_i,) we take the difference between the value of s at ℓ_i and its initial value at ℓ_0. Thus, we have

$$\delta_0 = \delta_1 = \delta_2 = \delta_6 = \delta_7 = 0,$$

since when P_1 is being executed alone the value of s at locations ℓ_0, ℓ_1, ℓ_2, ℓ_6, ℓ_7 is equal to the value of s at ℓ_0, *i.e.*, $s = 1$. Moreover,

$$\delta_3 = \delta_4 = \delta_5 = 1;$$

since when P_1 is executing alone, the value of s at locations ℓ_3, ℓ_4, ℓ_5 is smaller by 1 than the value of s at ℓ_0. Hence, the compensation expression for the body s in P_1 is

$$Z_1 = at\,\ell_3 + at\,\ell_4 + at\,\ell_5.$$

Computing the compensation expression Z_j for the body B for each process P_j we form the full invariant:

$$B + \sum_{j=1}^{m} Z_j \;=\; C.$$

For the three bodies we considered, we obtain the following three invariants:

$$I_1: \quad s + at\,\ell_{3..5} + at\,m_{2..5} \;=\; 1$$

$$I_2: \quad cf + ce + at\,\ell_{2..6} + at\,m_{1..6} \;=\; N$$

$$I_3: \quad |b| + ce + at\,\ell_{2..4} + at\,m_{5,6} \;=\; N.$$

Note that Q_3 can be obtained by forming the difference $I_2 - I_3$.

This method of deriving invariants has the advantage that no further proof is needed; indeed, any invariant derived by the method is automatically a true invariant of the program. But it may only be applied to variables which are modified by a constant in atomic instructions, or to programs which can be transformed so as to satisfy this restriction.

EXAMPLE: BINOMIAL COEFFICIENT

Consider next the program BC ([MP2]) for the distributed computation of the binomial coefficient $\binom{n}{k}$ for input parameters $n \geq k \geq 0$.

Program BC_1 (Binomial Coefficient first version)

$$y_1 := n, \quad y_2 := 0, \quad y_3 := 1, \quad y_4 := 1$$

```
$\ell_0$ : if $y_1 = (n-k)$ then go to $\ell_e$        $m_0$ : if $y_2 = k$ then go to $m_e$
$\ell_1$ : request($y_4$)                         $m_1$ : $y_2 := y_2 + 1$
|$\ell_2$ : $t_1 := y_3 \cdot y_1$ |                    $m_2$ : loop until $y_1 + y_2 \leq n$
|$\ell_3$ : $y_3 := t_1$         |                    $m_3$ : request($y_4$)
|$\ell_4$ : release($y_4$)       |                   |$m_4$ : $t_4 := y_3/y_2$ |
$\ell_5$ : $y_1 := y_1 - 1$                        |$m_5$ : $y_3 := t_2$     |
$\ell_6$ : go to $\ell_0$                          |$m_6$ : release($y_4$)   |
$\ell_e$ : halt                                  $m_7$ : go to $m_0$
                                              $m_e$ : halt

         $P_1$                                          $P_2$
```

The task of computing the binomial coefficient

$$\binom{n}{k} = \frac{n \cdot (n-1) \cdot \cdots \cdot (n-k+1)}{1 \cdot 2 \cdot \cdots \cdot k}$$

is distributed between the two processes by having P_1 perform all the multiplications while P_2 is in charge of the divisions. The values of y_1, *i.e.*, n, $n-1$, $\ldots$, $n-k+1$, are used to compute the numerator in P_1 (the last value of y_1, $n-k$, is not used), and the values of y_2, *i.e.*, 1, 2, $\ldots$, k, are used to compute the denominator (the first value of $y_2, 0$, is not used). The two processes must synchronize in order that the accumulated product be evenly divisible by the divisors used at m_4 by P_2. This synchronization is realized by the waiting loop at m_2 which essentially ensures that execution will proceed to m_3 only when at least y_2 factors have been multiplied into y_3. We rely here on the mathematical theorem that the product of i consecutive positive integers: $k \cdot (k+1) \cdot \cdots \cdot (k+i-1)$ is always divisible by $i!$. For, consider the intermediate expression at m_2:

$$y_3 = \frac{n \cdot (n-1) \cdot \cdots \cdot (n-j+1)}{1 \cdot 2 \cdot \cdots \cdot (i-1)},$$

where $1 \leq i \leq j \leq n$, $y_1 = n - j$ and $y_2 = i$. The numerator consists of a multiplication of i consecutive positive integers and it is therefore divisible by i. If $j = i$, we have to wait until y_1 is decremented by the instruction in ℓ_5 from $n-i+1$ to $n-i$ before we can be absolutely sure that $(n-i+1)$ has been multiplied into y_3. Thus, Process P_2 waits at m_2 until $y_1 + y_2$ drops to a value less than or equal to n.

The critical sections $L = \{\ell_2, \ell_3, \ell_4\}$ and $M = \{m_4, m_5, m_6\}$, protected by the semaphore variable y_4, ensure exclusive access to the shared variable y_3. Note that this program satisfies the single critical access rule ([MP2]) since for example in the expression $y_1 + y_2$ appearing at m_2 only y_1 is critically accessed.

The invariant

$$I_0: \quad at\,\ell_{2..4} + at\,m_{4..6} + y_4 \;=\; 1$$

ensures the mutual exclusion of the critical sections. It is verifiable by the invariance principle in the usual way.

Once this exclusion is established we can transform this program to a simpler program BC_2 such that there is a faithful correspondence between executions of BC_1 and executions of BC_2. This implies that the correctness of BC_1 will follow from that of BC_2.

Program BC_2 (Binomial Coefficient   second version)

```
                    $y_1 := n$,   $y_2 := 0$,   $y_3 := 1$

   $\ell_0$ : $if\ y_1 = (n-k)\ then\ go\ to\ \ell_e$        $m_0$ : $if\ y_2 = k\ then\ go\ to\ m_e$
   $\ell_1$ : $y_3 := y_3 \cdot y_1$                         $m_1$ : $y_2 := y_2 + 1$
   $\ell_2$ : $y_1 := y_1 - 1$                               $m_2$ : $loop\ until\ y_1 + y_2 \leq n$
   $\ell_3$ : $go\ to\ \ell_0$                               $m_3$ : $y_3 := y_3 / y_2$
   $\ell_e$ : $halt$                                         $m_4$ : $go\ to\ m_0$
                                                              $m_e$ : $halt$

              $P_1$                                                  $P_2$
```

Next we introduce two virtual variables:

$$y_1^* = \textit{if } at\,\ell_2 \textit{ then } y_1 - 1 \textit{ else } y_1$$

$$y_2^* = \textit{if } at\,m_{2,3} \textit{ then } y_2 - 1 \textit{ else } y_2.$$

The need for the virtual variables is similar to that of the compensation expressions discussed above. The main invariant on which the correctness of the program is based is I_3 below

$$y_3 = [n \cdot (n-1) \cdots (y_1^* + 1)] \,/\, [1 \cdot 2 \cdots y_2^*]$$

which ties together y_1, y_2 and y_3 (or their virtual versions). It is invariant in the sense that it is preserved after y_1, y_2 and y_3 has each been properly updated. However since the updating of y_1 and y_3 in P_1 for example cannot occur simultaneously, we define y_1^* which is the anticipated updated value of y_1 as soon as y_3 is updated at ℓ_1. Similarly, y_2^* differs from y_2 between the updating of y_2 and the updating of y_3 in P_2.

We use the following invariants:

$$I_1 : \quad [(n - k + at\,\ell_{1,2}) \le y_1 \le n] \;\wedge\; [0 \le y_2 \le (k - at\,m_1)]$$

$$I_2 : \quad at\,m_3 \supset (y_1 + y_2) \le n$$

$$I_3 : \quad y_3 = [n \cdot (n-1) \cdots (y_1^* + 1)] \,/\, [1 \cdot 2 \cdots y_2^*]$$

In I_3, the product of a zero number of terms evaluates to 1.

The initiality of I_1 to I_3 is easily verifiable.

The two parts of I_1 can be verified separately by considering the transitions $\ell_0 \to \ell_1$, $\ell_2 \to \ell_3$ and $m_0 \to m_1$, $m_1 \to m_2$ respectively.

To verify I_2 we observe that on entering m_3, $y_1 + y_2 \le n$ holds true. Any possible P_1 transition while P_2 is at m_3 can only decrease the value of $y_1 + y_2$.

Consider now the verification of I_3. The only relevant transitions are $\ell_1 \to \ell_2$ and $m_3 \to m_4$. Denoting the values of the variables after the transition by $y_1^{*\prime}$, $y_2^{*\prime}$, y_3' respectively, we obtain for $\ell_1 \to \ell_2$:

$$\begin{aligned} y_3 &= [n \cdot (n-1) \cdots (y_1^* + 1)] \,/\, [1 \cdot 2 \cdots y_2^*] \\ \Rightarrow\quad y_3 \cdot y_1 &= [n \cdot (n-1) \cdots (y_1^* + 1) \cdot y_1^*] \,/\, [1 \cdot 2 \cdots y_2^*] \\ & \qquad\qquad\qquad\qquad \text{as at } \ell_1, y_1 = y_1^* \\ \Rightarrow\quad y_3' &= [n \cdot (n-1) \cdots (y_1^{*\prime} + 1)] \,/\, [1 \cdot 2 \cdots y_2^*]. \end{aligned}$$

Similarly for the $m_3 \to m_4$ transition:

$$\begin{aligned} y_3 &= [n \cdot (n-1) \cdots (y_1^* + 1)] \,/\, [1 \cdot 2 \cdots y_2^*] \\ \Rightarrow\quad y_3 \,/\, y_2 &= [n \cdot (n-1) \cdots (y_1^* + 1)] \,/\, [1 \cdot 2 \cdots (y_2^* + 1)] \end{aligned}$$

as at $m_3, y_2 = y_2^* + 1$

$$\Rightarrow \quad y_3' \;=\; [n \cdot (n-1) \cdots (y_1^* + 1)] \,/\, [1 \cdot 2 \cdots y_2^{*\prime}].$$

The even divisibility of y_3 by y_2 at m_3 is ensured by the fact that by I_2 we have that

$$y_1^* \leq y_1 \leq n - y_2.$$

Thus the number of consecutive factors in the numerator of y_3 is at least y_2 which is evenly divisible by y_2!

PROVING EVENTUALITIES

Here we will consider general methodologies for proving properties of the form

$$\models P \supset \Diamond Q.$$

Many of the cases that we will study focus on a special kind of eventualities called *accessibility statement*. Its characteristic form is

$$at\,\ell \supset \Diamond\, at\,\ell'$$

guaranteeing that being at ℓ we will eventually reach ℓ'. In more general form it can appear as:

$$(at\,\ell \wedge \phi) \supset \Diamond(at\,\ell' \wedge \phi'),$$

where we associate a pre-condition ϕ with the visit at ℓ and a post-condition ϕ' with the visit at ℓ'. The Intermittent-Assertion Method (see [BUR], [MW]) uses this implication as the basic statement for reasoning. Many useful eventuality properties are representable in this form. In this discussion we assume that ℓ and ℓ' belong to the same process. It is however possible to consider generalizations in which this assumption may be relaxed.

Our approach for proving eventuality properties, called *proof by eventuality chains*, is based on establishing a chain of eventualities that by transitivity leads to the ultimate establishing of the desired goal (see also [OL]). The main transitivity argument used here is:

$$\models \phi_1 \supset \Diamond \phi_2 \text{ and } \models \phi_2 \supset \Diamond \phi_3 \quad \Rightarrow \quad \models \phi_1 \supset \Diamond \phi_3.$$

Some common techniques that we use in our proofs are:

- We split a situation into several subcases and pursue each case to its conclusion.
- To establish implications of the form

$$\models (\exists k.\phi(k)) \supset \Diamond \phi'$$

we use induction

$$\models \phi(0) \supset \phi' \text{ and } \models \forall n.[\phi(n) \supset \Diamond(\phi(n-1) \vee \phi')] \quad \Rightarrow \quad \models (\exists k.\phi(k)) \supset \Diamond \phi'.$$

- We frequently establish $\models \phi \supset \Diamond \phi'$ by contradiction: we assume $\phi \wedge \Box \sim\phi'$ and pursue the consequences of this assumption. If we succeed in showing

$$\models \; [\phi \wedge \Box \sim\phi'] \supset \textit{false},$$

then we will have established our desired result. This technique is particularly useful in the verification of a statement of the form

$$at\,\ell \supset \Diamond \sim at\,\ell$$

in concurrent systems. The reason for that is that by assuming $\Box\, at\,\ell$ we are momentarily (for the duration of the analysis) halting one of the processes at ℓ and have only to analyze the possible movements of the other processes. This usually results in a significant simplification.

We start by presenting an example with an informal proof of its correctness relative to accessibility.

EXAMPLE: MUTUAL EXCLUSION (DEKKER) INFORMAL PROOFS

As a first example, consider the solution to the mutual exclusion problem that was first given by Dekker and described in ([DIJ]). Here, we assume a shared variable t that may be modified by both processes and two private boolean variables y_1 and y_2, each being set only by its owning process but may be examined by the other.

Program DK (*Mutual Exclusion - Dekker's Solution*):

$$t := 1, \;\; y_1 := y_2 := F$$

```
ℓ0 : execute                          m0 : execute
ℓ1 : y1 := T                          m1 : y2 := T
ℓ2 : if(y2 = F) then go to ℓ7         m2 : if(y1 = F) then go to m7
ℓ3 : if(t = 1) then go to ℓ2          m3 : if(t = 2) then go to m2
ℓ4 : y1 := F                          m4 : y2 := F
ℓ5 : loop until (t = 1)               m5 : loop until (t = 2)
ℓ6 : go to ℓ1                         m6 : go to m1
┌───────────────┐                     ┌───────────────┐
│ℓ7 : t := 2    │                     │m7 : t := 1    │
│ℓ8 : y1 := F   │                     │m8 : y2 := F   │
└───────────────┘                     └───────────────┘
ℓ9 : go to ℓ0                         m9 : go to m0

   — P1 —                                — P2 —
```

The variable y_1 in process P_1 (and y_2 for P_2 respectively) is set to T at ℓ_1 to signal the intention of P_1 to enter its critical section at ℓ_7. Next P_1 tests at ℓ_2 if P_2 has any interest in entering its own critical section. This is tested by checking if $y_2 = T$. If $y_2 = F$, P_1 proceeds immediately to its critical section. If $y_2 = T$ we have a competition between the two processes on the access right to their critical sections. This competition is resolved by using the variable t (turn) that has the value 1 if in case of conflict P_1 has the higher priority and the value 2 if P_2 has the higher priority. If P_1 finds that $t = 1$ it knows it is its turn to insist and it leaves y_1 on and just loops between ℓ_2 and ℓ_3 waiting for y_2 to drop to F. If it finds that $t = 2$ it realizes it should yield to the other and consequently it turns y_1 off and enters a loop at ℓ_5, waiting for t to change to 1. It knows that as soon as P_2 exits its critical section it will set t to 1 so it will not be waiting forever. Once t has been detected to be 1, P_1 returns to the active competition at ℓ_2.

We will proceed to prove for this program both mutual exclusion and accessiblity. They are complementary properties in this case. The first assures that the two processes cannot simultaneously enter their respective critical sections. The second assures that once a process wishes to enter its critical section it will eventually get there.

Mutual exclusion

To prove mutual exclusion we show the joint invariance of the following three assertions:

$$Q_1 : \quad (y_1 = T) \quad \equiv \quad at\{\ell_2, \ell_3, \ell_4, \ell_7, \ell_8\}$$

$$Q_2 : \quad (y_2 = T) \quad \equiv \quad at\{m_2, m_3, m_4, m_7, m_8\}$$

$$Q_3 : \quad \sim at\{\ell_7, \ell_8\} \;\lor\; \sim at\{m_7, m_8\}.$$

That is,

$$\models \; \Box(Q_1 \land Q_2 \land Q_3),$$

where the initial condition is given by

$$at\,\ell_0 \;\land\; at\,m_0 \;\land\; (t = 1) \;\land\; (y_1 = y_2 = F).$$

The inductiveness of the first two assertions is easily checked by considering the different transitions in each of the processes. They certainly hold initially.

To show the invariance of Q_3 which is the statement of mutual exclusion consider the possible transitions that could potentially falsify this assertion.

One such transition is $\ell_2 \rightarrow \ell_7$ while $at\{m_7, m_8\}$. However by Q_2, $at\{m_7, m_8\}$ implies $y_2 = T$ so that the transition $\ell_2 \rightarrow \ell_7$ is disabled. Similarly for the transition $m_2 \rightarrow m_7$ while $at\{\ell_7, \ell_8\}$.

Accessibility

Accessibility in this program is given for P_1 (the case for P_2 is similar) by

$$\models \; at\,\ell_1 \supset \Diamond\, at\,\ell_7.$$

The process P_1 signals its wish to enter the critical section by moving from ℓ_0 to ℓ_1. We then would like to prove that it eventually reaches the critical section at ℓ_7.

In analyzing this program we have to interpret the *execute* instructions at ℓ_0 and m_0 as a non-critical section. Consequently we cannot assume that being at ℓ_0 we will eventually get to ℓ_1. Hence the transition graph representation of the *execute* instruction at ℓ_0 (and similarly at m_0) should be represented as:

$$\begin{tikzcd} \ell_0 \arrow[loop left, "\mathit{true} \rightarrow {[\,]}"] \arrow[r, "\mathit{true} \rightarrow {[\,]}"] & \ell_1 \end{tikzcd}$$

That is, there is a nondeterministic choice between staying at ℓ_0 and proceeding to ℓ_1.

We will proceed to prove

Theorem: $\models \mathit{at}\,\ell_1 \supset \Diamond\, \mathit{at}\,\ell_7$.

Here we will present an informal proof of the statement, followed by the justification of some of the steps used in the proof. Motivated by recurrent patterns in the informal proof we will then introduce proof principles that could be used to construct a formal version of the same proof.

The proof of the theorem consists of a sequence of lemmas.

Lemma A: $\models\ [\mathit{at}\,\ell_3 \wedge (t = 1)] \supset \Diamond\, \mathit{at}\,\ell_7$

Proof of Lemma A:

Assume to the contrary that P_1 never takes the $\ell_2 \rightarrow \ell_7$ transition; then henceforth

$$\Box[(\mathit{at}\,\ell_2 \vee \mathit{at}\,\ell_3) \wedge (t = 1)]$$

since the only instruction assigning to t a value different from 1 is at ℓ_7 and as long as $t = 1$ and the transition $\ell_2 \rightarrow \ell_7$ is not taken, P_1 is restricted to $\{\ell_2, \ell_3\}$.

Under this invariance assumption $\mathit{at}\{\ell_2, \ell_3\} \wedge (t = 1)$, let us check the locations of P_2.

case a: P_2 is at m_5. Then $y_2 = F$ and will stay so. By fairness P_1 must eventually get to ℓ_2 and in the next transition out of ℓ_2 must go to ℓ_7 (y_2 being F). Thus

$$\models\ \mathit{at}\,m_5 \supset \Diamond\, \mathit{at}\,\ell_7.$$

case b: P_2 is at m_4. Then by the fairness requirement it will eventually reach m_5 so that by case *a*

$$\models\ \mathit{at}\,m_4 \supset \Diamond\, \mathit{at}\,\ell_7.$$

case c: P_2 is at m_3. Then in the next transition out of m_3, t is still 1 so the m_4 branch must be taken. Consequently by case *b*

$$\models\ \mathit{at}\,m_3 \supset \Diamond\, \mathit{at}\,\ell_7.$$

case d: P_2 is at m_2. Then since, by Q_1, $(\mathit{at}\,\ell_2 \vee \mathit{at}\,\ell_3) \supset y_1 = T$, and since we assumed that P_1 is restricted to $\{\ell_2, \ell_3\}$, the next transition of P_2 will take us to m_3. Thus by case *c* also have

$$\models \ at\, m_2 \supset \Diamond\, at\, \ell_7.$$

case e: P_2 is at m_1. Then obviously eventually P_2 will reach m_2 so that by case *d* we have

$$\models \ at\, m_1 \supset \Diamond\, at\, \ell_7.$$

case f: P_2 is at m_6. Then eventually P_2 will get to m_1, so by case *e*

$$\models \ at\, m_6 \supset \Diamond\, at\, \ell_7.$$

case g: P_2 is at m_0. Then either it will stay in m_0 forever or eventually exit to m_1. In the case that it stays in m_0 forever we have by Q_2, $\Box(y_2 = F)$. Thus in the next transition out of ℓ_2 we must proceed to ℓ_7. Otherwise P_2 will eventually get to m_1 which by case *f* leads again to $at\, \ell_7$. Thus in any case

$$\models \ at\, m_0 \supset \Diamond\, at\, \ell_7.$$

case h: Obviously by fairness

$$\models \ (at\, m_7 \vee at\, m_8 \vee at\, m_9) \ \supset \ \Diamond\, at\, m_0,$$

so that by case g, any of these cases also leads to the eventual realization of $at\, \ell_7$.

Thus by analyzing all the possible values of π_2 in P_2 we showed that $at\, \ell_7$ is eventually realized in any of them. Consequently we have that

$$\models \ [at\, \ell_3 \ \wedge \ (t = 1)] \ \supset \ \Diamond\, at\, \ell_7.$$

which is the desired result of Lemma A. ∎

Lemma B: $\models \ [at\{\ell_3, \ldots, \ell_6\} \ \wedge \ (t = 2)] \ \supset \ \sim at\{m_8, m_9, m_0\}$

Proof of Lemma B:

Consider first the invariance of the following statement:

$$Q_4 : \quad (t = 2) \supset \sim at\, m_8.$$

The transitions which may possibly falsify this statement are:

- $\ell_7 \rightarrow \ell_8$ while P_2 is at m_8. However, due to Q_3, $at\, \ell_7 \wedge at\, m_8$ is an impossible situation.
- $m_7 \rightarrow m_8$ while $t = 2$, but the transition sets $t = 1$, so that Q_4 does hold after the transition.

Having established $\models \Box Q_4$ we proceed to establish $\models \Box Q_5$ where

$$Q_5 : \ [at\{\ell_3, \ldots, \ell_6\} \ \wedge \ (t = 2)] \ \supset \ \sim at\{m_9, m_0\}.$$

Let us investigate the transitions that could possibly falsify Q_5. The relevant transitions are:

- $\ell_2 \rightarrow \ell_3$ while $at\{m_9, m_0\}$. However by Q_2, $at\{m_9, m_0\}$ implies that $y_2 = F$ which disables this transition.
- $m_8 \rightarrow m_9$ while $t = 2$. However in view of Q_4 the situation $(t = 2) \wedge at\, m_8$ is impossible so that the transition is also impossible.

Taking the conjunction of Q_4 and Q_5 we can infer the result of Lemma B. ∎

Lemma C: $\models \; at\,\ell_5 \supset \Diamond\, at\,\ell_7$.

Proof of Lemma C:

If we are at ℓ_5 there are two possibilities. Either we will eventually get to ℓ_6 with $t = 1$ or we will stay forever in ℓ_5 with $t = 2$ continuously.

In the first case we proceed to ℓ_1 and reach ℓ_2. There we either enter ℓ_7 immediately or get to ℓ_3 with $t = 1$. The value of t will not change on the way since the only possible change of t from 1 to 2 is performed by P_1 at $\ell_7 \rightarrow \ell_8$. By lemma A, being at ℓ_3 with $t = 1$ ultimately leads to ℓ_7.

The other case is in which $\Box(t = 2 \wedge at\,\ell_5)$. By lemma B we have that $\Box(\sim at\{m_8, m_9, m_0\})$. Since $at\,\ell_5$ is permanently true so will be $y_1 = F$ by Q_1.

Consider now all the possible locations of π_2 in P_2 excluding m_8, m_9, and m_0:

$at\,m_7$ will eventually lead us to m_8 and turn t to 1.

$at\,m_2$ will lead us to m_7 since $y_1 = F$ and then to m_8.

$at\,m_3$ will lead us to m_2 since $t = 2$.

$at\,m_1$ leads to m_2.

$at\,m_6$ leads to m_1.

$at\,m_5$ will eventually lead to m_6, having $t = 2$.

$at\,m_4$ leads to m_5.

Consequently all the locations in P_2 eventually cause t to turn to 1 and P_1 will eventually get out of ℓ_5 and proceed to ℓ_3 with $t = 1$. Lemma A then establishes the desired result. ∎

We are ready now to prove the desired accessibility theorem, that $\models \; at\,\ell_1 \supset \Diamond\, at\,\ell_7$.

Proof of Theorem:

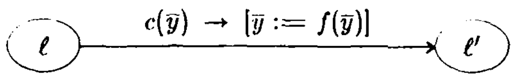
Proceed with P_1 from ℓ_1 to ℓ_2. There we either immediately enter ℓ_7 or arrive at ℓ_3. Consider the next instant in which P_1 is scheduled. If $t = 1$ we are assured by lemma A that we will ultimately get to ℓ_7. If $t = 2$ we proceed to ℓ_4 and ℓ_5 from which we are assured by lemma C of eventually getting to ℓ_7. Thus we will get to ℓ_7 in all cases. ∎

PROOF PRINCIPLES FOR EVENTUALITIES

In order to present proofs such as the above in a more rigorous perhaps even machine checkable – style, we proceed to develop several proof principles. These will enable us to establish the basic accessibility steps ensuring the eventual passage from a location to its successor under the assumption of fairness.

All predicates below are "state predicates" expressed by classical formulas, and will generally depend on the location variables $\bar{\pi}$ as well as on the program variables $\bar{y}$.

A predicate $\phi = \phi(\bar{\pi}; \bar{y})$ is said to be χ-*invariant*, where $\chi = \chi(\bar{\pi}; \bar{y})$, if for every transition



the following formula holds:

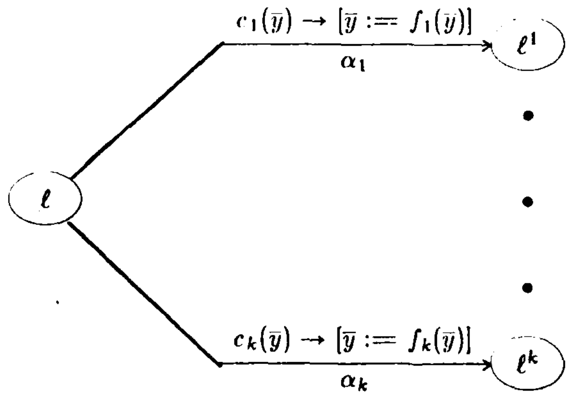
$$[at\,\ell \wedge c(\bar{y}) \wedge \chi(\bar{\pi}; \bar{y}) \wedge \chi(r(\bar{\pi}); f(\bar{y})) \wedge \phi(\bar{\pi}; \bar{y})] \supset \phi(r(\bar{\pi}); f(\bar{y})).$$

That is, ϕ is preserved by any transition which preserves χ.

In all the following we will use $\Box\chi$ to denote that χ is an invariant externally given and guaranteed to be continuously true. It will be useful in conducting conditional proofs.

The Escape Principle for Single Location

Consider a location ℓ in process P_j. Let $\Sigma = \{\alpha_1, \ldots, \alpha_k\}$ be a set of transitions originating in ℓ. Let $\ell^1, \ldots, \ell^k$ be the locations to which the transitions $\alpha_1, \ldots, \alpha_k$ lead and $c_1, \ldots, c_k$ the enabling conditions associated with $\alpha_1, \ldots, \alpha_k$, respectively. We do not require that Σ be the set of all transitions originating in ℓ.



We require that location ℓ be *deterministic*, that is, the conditions c and c' on any two distinct transitions α and α' (not necessarily in Σ) originating in ℓ must be disjoint, *i.e.* $\sim c \vee \sim c'$. In all the programs that we will study all locations would be deterministic except for those that contain an ***execute*** instruction. We will never apply the escape rule to these locations.

The Rule of Escape (ESC):

Let ϕ, χ, and ψ be predicates such that:

A: ϕ is $(at\,\ell \wedge \chi)$-invariant.
This means that as long as we stay at ℓ and χ is preserved, so is ϕ.

B: Any of the α_i, $i = 1, \ldots, k$, transitions of Σ that preserves χ and is initiated with ϕ true, achieves ψ, *i.e.*, ψ will hold after the transition. This is expressed by

$$[at\,\ell \ \wedge \ c_i(\overline{y}) \ \wedge \ \phi(\overline{\pi};\overline{y}) \ \wedge \ \chi(\overline{\pi};\overline{y}) \ \wedge \ \chi(r_i(\overline{\pi}); f_i(\overline{y}))] \ \supset \ \psi(r_i(\overline{\pi}); f_i(\overline{y}))$$

for every $i = 1, \ldots, k$.

C: $\phi \wedge \chi$ at ℓ ensures that at least one c_i, $i = 1, \ldots, k$, is true (the transition is enabled), *i.e.*,

$$[at\,\ell \ \wedge \ \phi(\overline{\pi};\overline{y}) \ \wedge \ \chi(\overline{\pi};\overline{y})] \ \supset \ \bigvee_{i=1}^{k} c_i(\overline{y}).$$

Then under these three conditions we may conclude

$$\models \ [at\,\ell \ \wedge \ \phi \ \wedge \ \Box\chi] \ \supset \ \Diamond\psi.$$

That is, being at ℓ with ϕ true and being assured of the continuous holding of χ guarantees eventual realization of ψ.

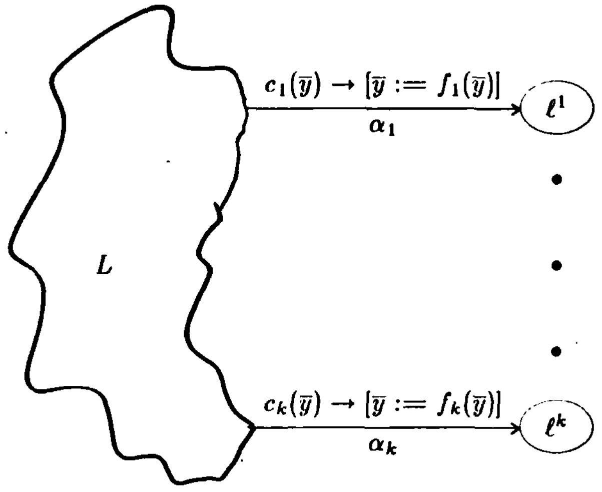
To justify the principle consider an execution which starts at ℓ with ϕ true and continuous assurance of χ. By condition A as long as P_j is not scheduled we remain at ℓ with $\phi \wedge \chi$ true. By condition C this implies that all that time $\bigvee_{i=1}^{k} c_i$ is also continuously true. Therefore by fairness eventually P_j must be scheduled in a state in which ϕ, χ, $\bigvee_{i=1}^{k} c_i$ all hold. Consequently by determinism of ℓ one of the $\alpha_i \in \Sigma$ transitions must be taken and by condition B, ψ must be realized.

There are some variations and generalizations of this basic principle which are discussed next.

The Rule of Alernatives for Regions

The first generalization considers exits out of a region (set of locations) rather than a single location. This principle applies also to nondeterministic locations.

Let $L \subseteq L_j$ be a set of locations in the process P_j and $\Sigma = \{\alpha_1, \ldots, \alpha_k\}$ the set of *all* transitions originating in L and leading to locations $\ell^1, \ldots, \ell^k$ outside of L, *i.e.*, $\ell^i \notin L$.



$c_1(\overline{y}) \to [\overline{y} := f_1(\overline{y})]$
α_1
ℓ^1
L
$c_k(\overline{y}) \to [\overline{y} := f_k(\overline{y})]$
α_k
ℓ^k

The Rule of Alternatives (ALT):

Let ϕ, χ, ψ be predicates such that:

A: ϕ is $(at\, L \wedge \chi)$ invariant.
This means that as long as we stay in L and χ is preserved so is ϕ.

B: Any of the $\alpha_i, i = 1, \ldots, k$, transitions of Σ that preserves χ and is initiated with ϕ true, achieves ψ, *i.e.*, ψ will hold after the transition. This is expressed by:

$$[at\, L \;\wedge\; c_i(\overline{y}) \;\wedge\; \phi(\overline{\pi};\overline{y}) \;\wedge\; \chi(\overline{\pi};\overline{y}) \;\wedge\; \chi(r_i(\overline{\pi}); f_i(\overline{y}))] \;\supset\; \psi(r_i(\overline{\pi}); f_i(\overline{y}))$$

for every $i = 1, \ldots, k$.

Then under these conditions we may conclude:

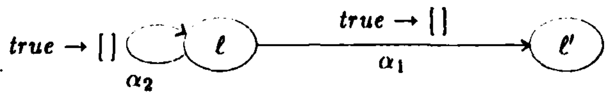
$$\models\; [at\, L \;\wedge\; \phi \;\wedge\; \Box\chi] \;\supset\; [\Box(at\, L \wedge \phi) \;\vee\; \Diamond\psi].$$

That is, being initially in L with ϕ true and being assured of the continuous holding of χ guarantees that we have two alternatives: either we stay in L with ϕ permanently true, or achieve ψ.

Note that since we do not have any condition similar to C above that guarantees the eventual realization of ψ, we must also consider the possibility of remaining in L and satisfying ϕ forever.

To justify the principle, consider an execution which starts in L with ϕ true and continuous assurance of χ. By condition A as long as we stay in L, ϕ will remain true. By condition B once we take any of the α_i transitions in this situation ψ will be realized. Hence the conclusion follows.

Note that the ALT rule can be applied to a region consisting of a single location. Thus for an *execute* instruction:

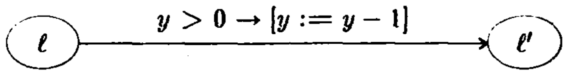


we may take $L = \{\ell\}$ and $\Sigma = \{\alpha_1\}$ to obtain

$$\models \; at\,\ell \supset [\Box\, at\,\ell \vee \Diamond\, at\,\ell'].$$

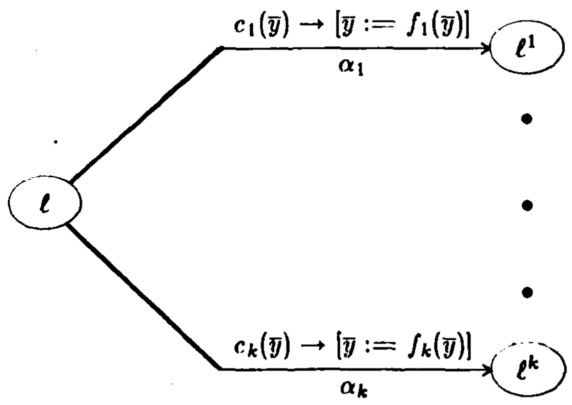
The Semaphore Rule

Rule ESC above is adequate for dealing with locations for which the disjunction of all their exit conditions (on all the outgoing transitions) is identically true. A location which does not satisfy this requirement is called a ***semaphore location*** since in a semaphore ***request*** instruction, represented by



the exit condition E_ℓ is $y > 0$ and is not identically true, nor is it necessarily continuously enabled. Consequently rules ESC and ALT are only sufficient for reasoning about programs that contain no sempahore locations. Once we have semaphore locations we need a stronger rule.

Let ℓ be a (possibly semaphore) location and $\Sigma = \{\alpha_1, \ldots, \alpha_k\}$ the set of *all* the transitions originating in ℓ. Let ℓ^i and c_i, for $i = 1, \ldots, k$, be respectively the location to which α_i leads and the condition enabling it.


$c_1(\bar{y}) \to [\bar{y} := f_1(\bar{y})]$
α_1
ℓ^1
ℓ
$c_k(\bar{y}) \to [\bar{y} := f_k(\bar{y})]$
α_k
ℓ^k

The Semaphore Rule (SEM):

Let ϕ, χ and ψ be state predicates such that:

A: ϕ is $(at\,\ell \wedge \chi)$-invariant.
This means that as long as we stay at ℓ and χ is preserved, so is ϕ.

B: Any of the $\alpha_i, i = 1, \ldots, k$, transitions of Σ, which preserves χ and is initiated with ϕ true, achieves ψ, *i.e.*, ψ will hold after the transition. This is expressed by:

$$[at\,\ell \wedge c_i(\bar{y}) \wedge \phi(\bar{\pi};\bar{y}) \wedge \chi(\bar{\pi};\bar{y}) \wedge \chi(r_i(\bar{\pi}); f_i(\bar{y}))] \supset \psi(r_i(\bar{\pi}); f_i(\bar{y}))$$

for every $i = 1, \ldots, k$.

C: If $(\phi \wedge \chi)$ holds permanently at ℓ then *eventually* one of the c_i, $i = 1, \ldots, k$, will be true. That is

$$\models \Box(at\,\ell \wedge \phi \wedge \chi) \supset \Diamond \bigvee_{i=1}^{k} c_i.$$

Then under these conditions we may conclude:

$$\models (at\,\ell \wedge \phi \wedge \Box\chi) \supset \Diamond\psi.$$

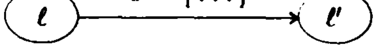
That is, being at ℓ with ϕ true and being assured of the continuous holding of χ guarantees the eventual realization of ψ.

Note that condition C of SEM is weaker than condition C of ESC in that it does not require $E_\ell = \bigvee_{i=1}^{k} c_i$ to be true whenever $at\,\ell \wedge \phi \wedge \chi$ holds but only requires it to be eventually realized. However, condition C here is a temporal statement and requires temporal reasoning for its justification, while condition C of ESC is static and requires only classical justification.

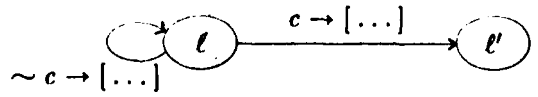
To justify this rule consider an execution which starts at ℓ with ϕ true and χ continuously maintained. Condition A ensures that as long as we stay at ℓ, $\phi \wedge \chi$ will be preserved. It is impossible that we stay at ℓ forever because by condition C this would imply that $E_\ell = \bigvee_{i=1}^{k} c_i$, which is the full exit condition of node ℓ, is enabled infinitely often while process P_j is never scheduled. By fairness we must have P_j scheduled at least once while E_ℓ is true. This, by condition B and the permanence until this moment of $\phi \wedge at\,\ell \wedge \chi$, will cause ψ to be realized.

It is important to realize the differences between a "semaphore location" and a "busy waiting" location. For comparison consider the following two simplified cases:

(*a*) ***Semaphore location:***



(*b*) ***Busy waiting location:***



(*a*) In the semaphore location case the fairness requirement demands that the scheduler will schedule this process at least once while its c condition is true provided the condition is true infinitely often. Thus for the SEM principle which is appropriate to this case we only require that c is realized infinitely often. This is exactly condition C which in this case is

$$\models \Box(at\,\ell \wedge \phi \wedge \chi) \supset \Diamond c,$$

or is equivalently

$$\models \Box(at\,\ell \wedge \phi \wedge \chi) \supset \Box\Diamond c.$$

(*b*) For the "busy waiting" situation, since the exit condition is $c \vee \sim c = \textit{true}$, the only obligation that the scheduler has is to eventually schedule this process. There is however nothing to prevent the process from being scheduled at exactly these instants in which c is false. Consequently, an infinitely often true c is not sufficient to ensure an exit to ℓ'. Instead we must require a stronger guarantee, that c be permanently true. Therefore, the corresponding condition C for the "busy waiting" situation for this case is

$$\models \quad (at\,\ell \wedge \phi \wedge \chi) \supset c,$$

which is equivalent to

$$\models \quad \Box(at\,\ell \wedge \phi \wedge \chi) \supset \Box c.$$

That is, if staying forever at ℓ guarantees the permanence of c then we will eventually exit from ℓ to ℓ'. This can be derived from the ESC rule.

Since $\models \Box c \supset \Diamond c$ we have the following ***robustness metatheorem***:

> A program that has been proven correct for an interpretation of its semaphores as "busy waiting" locations, is automatically correct for the implementation of these locations as true "semaphore" locations.

Consider, for example, the problem of accessibility of critical sections for the mutual exclusion program ME. In the roof to be given later we will reach the conclusion

$$\models \quad \Box\, at\,\ell_5 \supset \Diamond\Box(y_1 \neq y_2),$$

where the instruction at ℓ_5 is

$$\ell_5 : \quad \textit{loop while } y_1 = y_2.$$

Thus, this proof is sound for the interpretation of the *loop* primitive as "busy waiting". By the robustness metatheorem any more efficient implementation of the *loop* primitive, in fact any implementation at all which is "just", *i.e.*, eventually schedules each process, will also cause the program to behave correctly.

The Single Path Rule

In this derived rule we repetitively apply the ESC rule to a chain of locations.

Let $\ell_1, \ell_2, \ldots, \ell_{k+1}$ be a path of ***deterministic*** locations in P_j with an immediate transition α_i from every ℓ_i to ℓ_{i+1}, $i = 1, \ldots, k$.

$$\begin{tikzcd} \ell_1 \arrow[r, "c_1(y) \to [y := f_1(y)]", "\alpha_1"'] & \ell_2 \arrow[r] & \cdots & \ell_k \arrow[r, "c_k(y) \to [y := f_k(y)]", "\alpha_k"'] & \ell_{k+1} \end{tikzcd}$$

The Single Path Rule (SP):

Let χ, $\phi_1, \ldots, \phi_k$, and $\phi_{k+1} = \psi$ be predicates such that:

A: Each ϕ_i is $(at\,\ell_i \wedge \chi)$ invariant, $i = 1, \ldots, k$.
This means that as long as we stay at ℓ_i and χ is preserved so is ϕ_i.

B: Each transition α_i, $i = 1, \ldots, k$, which preserves χ and is initiated with ϕ_i true achieves ϕ_{i+1}, that is

$$[at\,\ell_i \wedge c_i(y) \wedge \phi_i(\pi; \bar{y}) \wedge \chi(\pi; y) \wedge \chi(r_i(\pi); f_i(y))] \supset \phi_{i+1}(r_i(\pi); f_i(y)).$$

C: $(\phi_i \wedge \chi)$ at ℓ_i ensures that c_i is true, *i.e.*,

$$[at\,\ell_i \wedge \phi_i \wedge \chi] \supset c_i.$$

Then under these three conditions we may conclude

$$\models [\bigvee_{i=1}^{k}(at\,\ell_i \wedge \phi_i) \wedge \Box\chi] \supset \Diamond\psi.$$

That is, if we start anywhere in the path with the appropriate ϕ_i true and χ continuously maintained *we eventually wind up having ψ.*

This rule is obviously a generalization of ESC and is justified by a repeated application of ESC to $\ell_1, \ldots, \ell_k$ (with $\Sigma_i = \{\alpha_i\}$) respectively.

This rule can be somewhat generalized to a more general graph than a path. The SP principle also applies instead to a tree in which every node has an edge directed towards its ancestor.

This concludes the list of semantic proof rules reflecting the structure of the program and its influence on the possible execution sequences.

* * * * *

In the following "formal" proofs of eventuality properties, we will intentionally omit manipulations which are pure temporal logic deductions, since we have not included an axiomatic system for temporal logic in this paper. Instead we will justify these deductions by saying "temporal reasoning" or "temporal deduction." The reader is invited to convince himself semantically that these deductions are indeed sound, that is, any sequence that satisfies the premises must also satisfy the consequence. Thus our ***proofs*** will consist, similarly to regular proofs, of a sequence of temporal formulas with a justification for each line in the sequence. A line in a proof may be justified in one of the following ways:

(*a*) If it is a valid first-order temporal logic formula.

(*b*) If it is an instance of one of the proof rules above.

(*c*) If it is a logical or temporal consequence of some preceding lines.

Given a deductive system for our logic (see [MAN2]) we will be able to justify steps of the form *b* and *c* using the axioms and rules of inference. Alternatively, *c* steps can be justified using a decision procedure for validity in (propositional) temporal logic ([BMP]). For our purpose of presenting proofs at a level which is not too formal, yet displays sufficient detail to be convincing, the style of semantic proofs seems most appropriate.

Note that our only reference to the program itself is through the proof principles ESC, ALT, SEM and SP.

In presenting formal (semantic) proofs we will work our way gradually through examples that use only the ESC and SP rules first, then examples that use also the ALT rule and finally examples using semaphores and the corresponding SEM rule.

EXAMPLE: COUNTING TREE NODES

Consider first the use of eventuality chains in proving the total correctness of the sequential program TN for counting the nodes of a binary tree.

Program TN (Counting the nodes of a tree):

```
        $S := (X),\ \ C := 0$
$\ell_0$ :  $if\ S = (\,)\ then\ goto\ \ell_e$
$\ell_1$ :  $(T, S) := (hd(S),\ tl(S))$
$\ell_2$ :  $if\ T = \Lambda\ then\ goto\ \ell_0$
$\ell_3$ :  $C := C + 1$
$\ell_4$ :  $S := \ell(T) \cdot r(T) \cdot S$
$\ell_5$ :  $goto\ \ell_0$
$\ell_e$ :  $halt.$
```

The program operates on a tree variable T and a variable S which is a stack of trees. The input variable X is a tree. The output is the value of the counter C. Each node in a tree may have zero, one or two descendants.

The available operations on trees are the functions $\ell(T)$ and $r(T)$ that yield the left and right subtrees of a tree T respectively. If the tree does not possess one of these subtrees the functions return the value Λ.

The stack S is initialized to contain the tree X. Taking the head and tail of a stack (functions *hd* and *tl* respectively) yields the top element and rest of the stack respectively. The operation in ℓ_1 pops the top of the stack into the variable T. The operation at ℓ_4 pushes both the right subtree and the left subtree of T onto the top of the stack.

At any iteration of the program, the stack S contains the list of subtrees of X whose nodes have not yet been counted. Each iteration removes one such subtree from the stack. If it is the empty subtree, $T = \Lambda$, we proceed to examine the next subtree on the stack. If it is not the empty subtree we add one to the counter C and pushes the left and right subtrees of T to the stack. When the stack is empty, $S = (\,)$, the program halts.

Denoting by $|X|$ the number of nodes in the tree X, the statement to be proved is formulated as

Theorem: $\models \; at\,\ell_0 \supset \Diamond(at\,\ell_e \wedge C = |X|)$.

In order to prove the theorem we first prove a lemma:

Lemma: $\models \; [at\,\ell_0 \;\wedge\; S = t \cdot s \;\wedge\; C = c] \;\supset\; \Diamond[at\,\ell_0 \;\wedge\; S = s \;\wedge\; C = c + |t|]$.

The lemma states that being at ℓ_0 with a tree t at the top of the stack S, we are assured of a later visit at ℓ_0 where t has been removed from the stack and its node count $|t|$ has been added to C.

Denote by $E(n)$ the statement:

$$E(n): \quad \forall t, s, c\, \{[at\,\ell_0 \;\wedge\; S = t \cdot s \;\wedge\; C = c \;\wedge\; |t| \leq n] \;\supset$$
$$\Diamond[at\,\ell_0 \;\wedge\; S = s \;\wedge\; C = c + |t|]\}.$$

This statement is the restriction of the lemma to trees with node count not exceeding n for some natural number $n \geq 0$.

Proof of Lemma:

The lemma can then be stated as $\models \forall n.\; E(n)$; it is proved by induction. We have to show

(a) $\models E(0)$

(b) $\models E(n) \supset E(n+1)$.

(a) Since $t \cdot s \neq (\,)$ and $|t| = 0 \supset t = \Lambda$ we may apply the SP rule to the path $\ell_0 \to \ell_1 \to \ell_2 \to \ell_0$ and obtain

$$\textbf{1.} \quad \models [at\,\ell_0 \;\wedge\; S = t \cdot s \;\wedge\; C = c \;\wedge\; |t| = 0] \;\supset$$
$$\Diamond[at\,\ell_0 \;\wedge\; S = s \;\wedge\; C = c].$$

This establishes $\models E(0)$.

(b) To show $\models E(n) \supset E(n+1)$, consider an arbitrary $n, n \geq 0$, and assume

$$\textbf{2.} \quad \models E(n).$$

Then

$$\textbf{3.} \quad \models [at\,\ell_0 \;\wedge\; S = t' \cdot s' \;\wedge\; C = c' \;\wedge\; |t'| = n+1] \;\supset$$
$$\Diamond[at\,\ell_0 \;\wedge\; S = \ell(t') \cdot r(t') \cdot s' \;\wedge\; C = c' + 1 \;\wedge\; |t'| = n+1]$$

by the SP rule applied to the path $\ell_0 \to \ell_1 \to \ell_2 \to \ell_3 \to \ell_4 \to \ell_5 \to \ell_0$, using $|t'| = n+1 \supset t' \neq \Lambda$.

We now use an instantiation of $E(n)$ with $t = \ell(t')$, $s = r(t') \cdot s'$, and $c = c' + 1$ (which is justified since $|t| = |\ell(t')| < n + 1$) to obtain

$$\mathbf{4.} \quad \models [at\,\ell_0 \wedge S = \ell(t') \cdot r(t') \cdot s' \wedge C = c' + 1] \supset \Diamond[at\,\ell_0 \wedge S = r(t') \cdot s' \wedge C = c' + 1 + |\ell(t')|].$$

By 3 and 4 we have

$$\mathbf{5.} \quad \models [at\,\ell_0 \wedge S = t' \cdot s' \wedge C = c' \wedge |t'| = n+1] \supset \Diamond[at\,\ell_0 \wedge S = r(t') \cdot s' \wedge C = c' + 1 + |\ell(t')| \wedge |t'| = n+1].$$

We now apply an instance of $E(n)$ again, this time with $t = r(t')$, $s = s'$, and $c = c' + 1 + |\ell(t')|$ (which is justified since $|t| = |r(t')| < n + 1$) to obtain

$$\mathbf{6.} \quad \models [at\,\ell_0 \wedge S = r(t') \cdot s' \wedge C = c' + 1 + |\ell(t')|] \supset \Diamond[at\,\ell_0 \wedge S = s' \wedge C = c' + 1 + |\ell(t')| + |r(t')|].$$

By 5 and 6 we have

$$\mathbf{7.} \quad \models [at\,\ell_0 \wedge S = t' \cdot s' \wedge C = c' \wedge |t'| = n+1] \supset \Diamond[at\,\ell_0 \wedge S = s' \wedge C = c' + 1 + |\ell(t')| + |r(t')|].$$

Using the property

$$|t| > 0 \Rightarrow |t| = 1 + |\ell(t)| + |r(t)|$$

we obtain:

$$\mathbf{8.} \quad \models [at\,\ell_0 \wedge S = t' \cdot s' \wedge C = c' \wedge |t'| = n+1] \supset \Diamond[at\,\ell_0 \wedge S = s' \wedge C = c' + |t'|].$$

Universally quantifying over the variables t', s' and c' and then renaming them to t, s and c, respectively, we obtain

$$\mathbf{9.} \quad \models \forall t, s, c\,\{[at\,\ell_0 \wedge S = t \cdot s \wedge C = c \wedge |t| = n+1] \supset \Diamond[at\,\ell_0 \wedge S = s \wedge C = c + |t|]\}.$$

Line 9 holds under assumption 2 for every $n, n \geq 0$. Combined with 1 this gives

$$\mathbf{10.} \quad E(n) \models E(n+1).$$

Therefore, by the deduction theorem we have

11. $\models E(n) \supset E(n+1)$.

This concludes the proof of the lemma.

Proof of Theorem:

To prove the theorem we observe that

12. $\models [at\,\ell_0 \wedge S = (X) \wedge C = 0] \supset \Diamond[at\,\ell_0 \wedge S = (\,) \wedge C = |X|]$

by the lemma with $t = X$, $s = (\,)$, and $c = 0$. But

13. $\models [at\,\ell_0 \wedge S = (\,) \wedge C = |X|] \supset \Diamond[at\,\ell_e \wedge C = |X|]$

by SP applied to $\ell_0 \rightarrow \ell_e$. Therefore, by combining 12 and 13, we have

14. $\models [at\,\ell_0 \wedge S = (X) \wedge C = 0] \supset \Diamond[at\,\ell_e \wedge C = |X|]$

i.e.,

15. $\models \Diamond[at\,\ell_e \wedge C = |X|]$. ▮

One cannot fail to see the close resemblance between the temporal proof presented here and the informal incrmittent-assertion proof conducted in [BUR] and [MW]. Our *SP* principle replaces the "little hand simulation" of [BUR].

EXAMPLE: MUTUAL EXCLUSION (DEKKER) FORMAL PROOFS

We will now present a formal proof of the accessibility proof of the program DK. An informal proof of this was presented before and we advise the reader to refer to it while reading the following proof. The accessibility statement to be proved is

Theorem: $\models at\,\ell_1 \supset \Diamond\, at\,\ell_7$.

We will make use of the invariants derived before, namely:

$$\models \Box(Q_1 \wedge Q_2 \wedge Q_3 \wedge Q_4)$$

where

$$Q_1: \quad (y_1 = T) \equiv at\{\ell_2, \ell_3, \ell_4, \ell_7, \ell_8\}$$

$$Q_2: \quad (y_2 = T) \equiv at\{m_2, m_3, m_4, m_7, m_8\}$$

$$Q_3: \quad \sim at\{\ell_7, \ell_8\} \vee \sim at\{m_7, m_8\}$$

and

$$Q_4: \quad [at\{\ell_3, \ldots, \ell_6\} \wedge (t = 2)] \supset \sim at\{m_8, m_9, m_0\}.$$

Q_4 was proved by the standard invariance rule in Lemma B and will not be reproven here.

The proof of the theorem consists of a sequence of lemmas.

Lemma A: $\models [at\,\ell_{2,3} \wedge (t = 1)] \supset \Diamond at\,\ell_7$

Proof of Lemma A:

1. $\models [at\,\ell_{2,3} \wedge (t = 1)] \supset \{\Box[at\,\ell_{2,3} \wedge (t = 1)] \vee \Diamond at\,\ell_7\}$

by the ALT rule at $\ell_{2,3}$ where ϕ is $t = 1$. Note that by $t = 1$, the $\ell_3 \rightarrow \ell_4$ transition is never possible.

2. $\models [at\,\ell_{2,3} \wedge (t = 1) \wedge at\,m_5] \supset [at\,\ell_{2,3} \wedge (t = 1) \wedge at\,m_5 \wedge (y_2 = F)]$

by Q_2.

3. $\models [at\,\ell_{2,3} \wedge (t = 1) \wedge at\,m_5 \wedge (y_2 = F)] \supset \Diamond at\,\ell_7$

by SP applied to the path $\ell_3 \rightarrow \ell_2 \rightarrow \ell_7$ where $\phi_3 = \phi_2$ is $(t = 1) \wedge at\,m_5 \wedge (y_2 = F)$ and ψ is $at\,\ell_7$.

4. $\models \{\Box[at\,\ell_{2,3} \wedge (t = 1)] \wedge at\,m_5\} \supset \Diamond at\,\ell_7$

is a temporal conclusion of 2 and 3.

This corresponds to case *a* of Lemma A in the informal proof.

Next we have

5. $\models \Box[at\,\ell_{2,3} \wedge (t = 1)] \supset \Box[at\,\ell_{2,3} \wedge (t = 1) \wedge (y_1 = T)]$

by Q_1.

6. $\models \{\Box[at\,\ell_{2,3} \wedge (t = 1) \wedge (y_1 = T)] \wedge at\{m_{1..4}; m_6\}\} \supset \Diamond at\,m_5$

by the SP rule applied to the path $m_6 \rightarrow m_1 \rightarrow m_2 \rightarrow m_3 \rightarrow m_4 \rightarrow m_5$ where χ is $at\,\ell_{2,3} \wedge (t = 1) \wedge (y_1 = T)$.

7. $\models \{\Box[at\,\ell_{2,3} \wedge (t = 1)] \wedge at\{m_{1..4}, m_6\}\} \supset \Diamond at\,m_5$

by 5 and 6.

8. $\models \{\Box[at\,\ell_{2,3} \wedge (t = 1)] \wedge at\,m_{1..6}\} \supset \Diamond at\,\ell_7$

by 7 and 4.

This covers cases *b, c, d, e, f* of the informal Lemma *A*.

We have

$$9.\quad \models\ at\, m_0 \ \supset\ [at\, m_0 \wedge (y_2 = F)]$$

by Q_2.

$$10.\quad \models\ [at\, m_0 \wedge (y_2 = F)] \ \supset\ \{\Box[at\, m_0 \wedge (y_2 = F)] \vee \Diamond\, at\, m_1\}$$

by ALT at m_0 where ϕ is $y_2 = F$. Therefore

$$11.\quad \models\ at\, m_0 \ \supset\ [\Box(y_2 = F) \vee \Diamond\, at\, m_1]$$

by 9 and 10.

$$12.\quad \models\ [\Box(y_2 = F) \wedge at\, \ell_{2,3} \wedge (t = 1)] \ \supset\ \Diamond\, at\, \ell_7$$

by the SP rule applied to $\ell_3 \to \ell_2 \to \ell_7$ where $\phi_3 = \phi_2$ is $t = 1$ and χ is $y_2 = F$.

$$13.\quad \models\ \{\Box[at\, \ell_{2,3} \wedge (t = 1)] \wedge \Box(y_2 = F)\} \ \supset\ \Diamond\, at\, \ell_7$$

is a consequence of 12. By taking the disjunction of 13 and 8 we get

$$14.\quad \models\ \{\Box[at\, \ell_{2,3} \wedge (t = 1)] \wedge \big(\Box(y_2 = F) \vee at\, m_{1..6}\big)\} \ \supset\ \Diamond\, at\, \ell_7$$

and then

$$15.\quad \models\ \{\Box[at\, \ell_{2,3} \wedge (t = 1)] \wedge at\, m_0\} \ \supset\ \Diamond\, at\, \ell_7$$

is a consequence of 11 and 14.

This covers case *g* of the informal Lemma *A*.

We also have

$$16.\quad \models\ \{\Box[at\, \ell_{2,3} \wedge (t = 1)] \wedge at\, m_{7..9}\} \ \supset\ \Diamond\, at\, m_0$$

by the SP rule applied to the path $m_7 \to m_8 \to m_9 \to m_0$.

$$17.\quad \models\ \{\Box[at\, \ell_{2,3} \wedge (t = 1)] \wedge at\, m_{7..9}\} \ \supset\ \Diamond\, at\, \ell_7$$

by 15 and 16.

This covers case *h* of the proof.

Taking the disjunction of 8, 15 and 17 we obtain

$$18.\quad \models\ \Box[at\, \ell_{2,3} \wedge (t = 1)] \supset\ \Diamond\, at\, \ell_7.$$

Taking together 1 and 18 yields

$$\textbf{19.} \quad \models \ [at\,\ell_{2,3} \ \wedge \ (t=1)] \ \supset \ \Diamond \, at\,\ell_7$$

which is the result of Lemma A.

Lemma B is an invariance property $\models Q_4$ and is proved using the invariance principle.

Lemma C: $\models \ at\,\ell_5 \supset \Diamond \, at\,\ell_7$

Proof of Lemma C:

$$\textbf{1.} \quad \models \ at\,\ell_5 \ \supset \ \{\Box \, at\,\ell_5 \ \vee \ \Diamond[at\,\ell_6 \wedge (t=1)]\}$$

by the ALT rule at ℓ_5.

$$\textbf{2.} \quad \models \ \Box(t=2) \ \vee \ \Diamond(t=1)$$

is a temporal tautology using the obvious invariance $(t=1) \vee (t=2)$.

$$\textbf{3.} \quad \models \ \Box \, at\,\ell_5 \ \supset \ \{\Box[at\,\ell_5 \ \wedge \ (t=2)] \ \vee \ \Diamond[at\,\ell_5 \wedge (t=1)]\}$$

is a temporal consequence of 2.

$$\textbf{4.} \quad \models \ [at\,\ell_5 \ \wedge \ (t=1)] \ \supset \ \Diamond[at\,\ell_6 \wedge (t=1)]$$

by the ESC rule at ℓ_5 where ϕ is $t=1$.

$$\textbf{5.} \quad \models \ \Box \, at\,\ell_5 \ \supset \ \{\Box[at\,\ell_5 \wedge (t=2)] \ \vee \ \Diamond[at\,\ell_6 \wedge (t=1)]\}$$

is a temporal consequence of 3 and 4.

$$\textbf{6.} \quad \models \ at\,\ell_5 \ \supset \ \{\Box[at\,\ell_5 \wedge (t=2)] \ \vee \ \Diamond[at\,\ell_6 \wedge (t=1)]\}$$

by 1 and 5.

$$\textbf{7.} \quad \models \ \Box[at\,\ell_5 \wedge (t=2)] \ \supset \ \Box[at\,\ell_5 \wedge (t=2) \wedge (y_1=F) \wedge at\,m_{1..7}]$$

by Q_1 and Q_4.

We have

$$\textbf{8.} \quad \models \ \{\Box[at\,\ell_5 \ \wedge \ (t=2)] \ \wedge \ at\,m_7\} \ \supset \ \Diamond[at\,\ell_5 \wedge (t=1)]$$

by the ESC rule at m_7 where χ is $at\,\ell_5 \wedge (t=2)$, ψ is $at\,\ell_5 \wedge (t=1)$.

$$\textbf{9.} \quad \models \ \{\Box[at\,\ell_5 \ \wedge \ (t=2)] \ \wedge \ at\,m_7\} \ \supset \ \Diamond[at\,\ell_6 \wedge (t=1)]$$

by 8 and 4.

This covers case a of the informal Lemma C.

Denoting

$$\chi_0 : \quad at\,\ell_5 \;\wedge\; (t_2 = 2) \;\wedge\; (y_1 = F) \;\wedge\; at\,m_{1..7}$$

we have

10. $\models \; [\Box\chi_0 \;\wedge\; at\{m_{1,2}, m_{4..7}\}] \;\supset\; \Diamond[\chi_0 \;\wedge\; at\,m_7]$

by the SP rule applied to the path $m_4 \rightarrow m_5 \rightarrow m_6 \rightarrow m_1 \rightarrow m_2 \rightarrow m_7$.

11. $\models \; [\Box\chi_0 \;\wedge\; at\{m_{1,2}, m_{4..7}\}] \;\supset\; \Diamond[at\,\ell_6 \;\wedge\; (t = 1)]$

by 10 and 9.

This covers cases b, d, e, f, g of the informal Lemma C.

We have

12. $\models \; [\Box\chi_0 \;\wedge\; at\,m_3] \;\supset\; \Diamond\, at\,m_2$

by the ESC rule at m_3. Thus

13. $\models \; [\Box\chi_0 \;\wedge\; at\,m_3] \;\supset\; \Diamond[at\,\ell_6 \;\wedge\; (t = 1)]$

by 11 and 12.

This covers case c of the informal Lemma C.

Taking the disjunction of 11 and 13 and noting that $\chi_0 \supset at\,m_{1..7}$ we obtain

14. $\models \; \Box\chi_0 \;\supset\; \Diamond[at\,\ell_6 \wedge (t = 1)]$.

Combined with 7 this gives

15. $\models \; \Box[at\,\ell_5 \wedge (t = 2)] \;\supset\; \Diamond[at\,\ell_6 \wedge (t = 1)]$.

Combined with 6 we obtain

16. $\models \; at\,\ell_5 \;\supset\; \Diamond[at\,\ell_6 \wedge (t = 1)]$.

Now we can derive

17. $\models \; [at\,\ell_{1,6} \;\wedge\; (t = 1)] \;\supset\; \Diamond[at\,\ell_{2,3} \wedge (t = 1)]$

by the SP rule applied to the path $\ell_6 \rightarrow \ell_1 \rightarrow \ell_2$ where $\phi_6 = \phi_1$ is $(t = 1)$, ψ is $at\,\ell_{2,3} \wedge (t = 1)$. Using now Lemma A we obtain

18. $\models \; [at\,\ell_{1,6} \;\wedge\; (t = 1)] \;\supset\; \Diamond\, at\,\ell_7$

which together with 16 gives

19. $\models \; at\,\ell_5 \supset \Diamond\, at\,\ell_7$.

Proof of Theorem:

Consider now the final proof of the theorem

1. $\models \; at\,\ell_1 \supset \Diamond\, at\,\ell_2$ — by ESC rule at ℓ_1
2. $\models \; at\,\ell_2 \supset \Diamond[at\,\ell_7 \vee at\,\ell_3]$ — by the ESC rule at ℓ_2
3. $\models \; at\,\ell_2 \supset [\Diamond\, at\,\ell_7 \vee \Diamond\, at\,\ell_3]$ — which is temporally equivalent to 2
4. $\models \; at\,\ell_3 \supset \{\Diamond[at\,\ell_2 \wedge (t=1)] \vee \Diamond\, at\,\ell_4\}$ — by the ESC rule at ℓ_3
5. $\models \; [at\,\ell_2 \wedge (t=1)] \supset \Diamond\, at\,\ell_7$ — by Lemma A
6. $\models \; at\,\ell_4 \supset \Diamond\, at\,\ell_5$ — by ESC rule at ℓ_4
7. $\models \; at\,\ell_4 \supset \Diamond\, at\,\ell_7$ — by Lemma C and 6
8. $\models \; at\,\ell_3 \supset \Diamond\, at\,\ell_7$ — by 4, 5, and 7
9. $\models \; at\,\ell_2 \supset \Diamond\, at\,\ell_7$ — by 3 and 8
10. $\models \; at\,\ell_1 \supset \Diamond\, at\,\ell_7$ — by 1 and 9

This concludes the proof of the theorem.

EXAMPLE: CONSUMER PRODUCER

Consider next proving accessibility for the Consumer-Producer program (program CP). We assume that the computations at ℓ_0 and at m_7 eventually terminate. The statement to be proved is:

Theorem: $\models \; at\,\ell_0 \supset \Diamond\, at\,\ell_3$

We will use in our proof the invariants which were established before

$$\models \; \Box(Q_0 \wedge Q_1 \wedge Q_2)$$

where

$$Q_0 : \; (cf \geq 0) \wedge (ce \geq 0) \wedge (s \geq 0)$$
$$Q_1 : \; at\,\ell_{3..5} + at\,m_{2..5} + s \; = \; 1$$
$$Q_2 : \; cf + ce + at\,\ell_{2..6} + at\,m_{1..6} \; = \; N$$

Note that this is the first example that uses semaphores.

Assuming that the computation of y_1 at ℓ_0 eventually terminates we may conclude

$$\models \; at\,\ell_0 \supset \Diamond\, at\,\ell_1.$$

The rest of the theorem is proved by two lemmas. Lemma A ensures that we get from ℓ_1 to ℓ_2 and Lemma B ensures that we get from ℓ_2 to ℓ_3.

Lemma A: $\models \; at\,\ell_1 \supset \Diamond\, at\,\ell_2$

Proof of Lemma A:

Since location ℓ_1 contains a semaphore ***request*** instruction we will use the semaphore rule SEM to show that eventually P_1 will be granted access to ℓ_2. The premise needed for the SEM rule is $\Box\, at\,\ell_1 \supset \Diamond(ce > 0)$. An intuitive interpretation of this premise is that if we wait long enough at ℓ_1, ce will eventually turn positive. To show this, we give first an informal exposition inspecting the different locations in which P_2 may currently be.

case a: P_2 is at m_6. Then eventually it will execute the *release*(ce) instruction to get $ce > 0$ as required.

case b: P_2 is at m_2, m_3, m_4 or m_5. Then it will eventually get to m_6 which by case a will cause ce to turn positive.

case c: P_2 is at m_1. Then since P_1 is at ℓ_1, $s = 1$ by Q_1. Since we assume that P_1 is waiting at ℓ_1, s will remain 1 as long as P_2 stays at m_2. By the semaphore axiom applied at m_1, P_2 will eventually proceed to m_2 and by case b, ce will eventually turn positive.

case d: P_2 is at m_0. Then since P_1 is at ℓ_1, $cf + ce = N > 0$ by Q_2. If $ce > 0$ we have proven our claim. Otherwise $cf > 0$ and will remain so as long as P_2 stays at m_0. Again by the semaphore axiom P_2 must eventually advance to m_1 and then by case c, ce will eventually turn positive.

case e: P_2 is at m_7 or m_8. It will eventually get to m_0 and then by case d, ce will eventually turn positive.

Let us now proceed with the more formal proof:

1. $\models \; [\Box\, at\,\ell_1 \wedge at\,m_6] \supset [\Box\, at\,\ell_1 \wedge at\,m_6 \wedge (ce \geq 0)]$

by Q_0.

2. $\models \; [\Box\, at\,\ell_1 \wedge at\,m_6 \wedge (ce \geq 0)] \supset \Diamond(ce > 0)$

by ESC applied at m_6 where ϕ is $ce \geq 0$, χ is $at\,\ell_1$, ψ is $ce > 0$.

3. $\models \; [\Box\, at\,\ell_1 \wedge at\,m_6] \supset \Diamond(ce > 0)$

is a conclusion of 1 and 2.

This corresponds to case **a** above.

We have

4. $\models \; [\Box\, at\,\ell_1 \wedge at\,m_{2..5}] \supset \Diamond\, at\,m_6$

by the SP rule applied to the path $m_2 \rightarrow m_3 \rightarrow m_4 \rightarrow m_5 \rightarrow m_6$.

$$5.\quad \models\ [\Box\, at\,\ell_1\ \wedge\ at\,m_{2..5}]\ \supset\ \Diamond(ce > 0)$$

is a conclusion of 4 and 3.

This covers case *b* above.

We have

$$6.\quad \models\ [at\,\ell_1\ \wedge\ at\,m_1]\ \supset\ (s = 1)$$

by Q_1.

$$7.\quad \models\ [\Box\, at\,\ell_1\ \wedge\ \Box\, at\,m_1]\ \supset\ \Diamond(s = 1)$$

is a temporal consequence of 6.

$$8.\quad \models\ [\Box\, at\,\ell_1\ \wedge\ at\,m_1]\ \supset\ \Diamond\, at\,m_2$$

by the SEM rule at m_1 where χ is $at\,\ell_1$.

$$9.\quad \models\ [\Box\, at\,\ell_1\ \wedge\ at\,m_1]\ \supset\ \Diamond(ce > 0)$$

is a conclusion of 8 and 5.

This covers case *c*.

We have

$$10.\quad \models\ [\Box\, at\,\ell_1\ \wedge\ at\,m_0]\ \supset\ [(cf > 0)\ \vee\ (ce > 0)]$$

by Q_2.

$$11.\quad \models\ \Box(at\,\ell_1 \wedge at\,m_0 \wedge (cf > 0))\ \supset\ \Diamond(cf > 0)$$

is a trivial temporal tautology.

$$12.\quad \models\ [\Box\, at\,\ell_1\ \wedge\ at\,m_0\ \wedge\ (cf > 0)]\ \supset\ \Diamond\, at\,m_1$$

by the SEM rule at m_0, where ϕ is $cf > 0$, χ is $at\,\ell_1$.

$$13.\quad \models\ [\Box\, at\,\ell_1\ \wedge\ at\,m_0\ \wedge\ (cf > 0)]\ \supset\ \Diamond(ce > 0)$$

is a conclusion of 12 and 9.

$$14.\quad \models\ [\Box\, at\,\ell_1\ \wedge\ at\,m_0]\ \supset\ \Diamond(ce > 0)$$

by a disjunction of 10 and 13.

This corresponds to case *d*.

We have

15. $\models$ $[\Box\, at\,\ell_1 \;\wedge\; at\,m_{7,8}] \;\supset\; \Diamond\, at\,m_0$

by the SP rule applied to the path $\ell_7 \to \ell_8 \to \ell_0$.

16. $\models$ $[\Box\, at\,\ell_1 \;\wedge\; at\,m_{7,8}] \;\supset\; \Diamond(ce > 0)$

by 15 and 14.

This covers case *e*.

By taking the disjunction of 3, 5, 9, 14 and 16 we obtain

17. $\models$ $\Box\, at\,\ell_1 \;\supset\; \Diamond(ce > 0)$.

By applying the SEM rule at ℓ_1 we obtain

18. $\models$ $at\,\ell_1 \;\supset\; \Diamond\, at\,\ell_2$. ∎

Lemma B: $\models$ $at\,\ell_2 \supset \Diamond\, at\,\ell_3$

Proof of Lemma B:

Here again we will apply the SEM rule, this time at ℓ_2. The needed premise for its application is:

$\models$ $\Box\, at\,\ell_2 \;\supset\; \Diamond(s > 0)$.

By inspecting the current location of P_2 we distinguish three cases:

case a: P_2 is at m_5. It will eventually advance to m_6 and turn s positive.

case b: P_2 is somewhere in $\{m_2, m_3, m_4\}$. It will eventually get to m_5 and then by case *a* will turn s positive.

case c: P_2 is somewhere in $\{m_0, m_1, m_6, m_7, m_8\}$. By Q_1, since P_1 is at ℓ_1, s is currently equal to 1.

Thus the more formal proof is given by:

1. $\models$ $[\Box\, at\,\ell_2 \;\wedge\; at\,m_5] \;\supset\; [\Box\, at\,\ell_2 \;\wedge\; at\,m_5 \;\wedge\; (s \geq 0)]$

by Q_0.

2. $\models$ $[\Box\, at\,\ell_2 \;\wedge\; at\,m_5 \;\wedge\; (s \geq 0)] \;\supset\; \Diamond(s > 0)$

by ESC applied at m_5 where ϕ is $s \geq 0$, χ is $at\,\ell_2$, ψ is $s > 0$

3. $\models$ $[\Box\, at\,\ell_2 \;\wedge\; at\,m_5] \;\supset\; \Diamond(s > 0)$

is a conclusion of 1 and 2.

This covers case *a*.

We have

$$4. \quad \models \; [\Box\, at\,\ell_2 \;\wedge\; at\,m_{2..4}] \;\supset\; \Diamond(at\,m_5)$$

by the SP rule applied to the path $m_2 \rightarrow m_3 \rightarrow m_4 \rightarrow m_5$.

$$5. \quad \models \; [\Box\, at\,\ell_2 \;\wedge\; at\,m_{2..4}] \;\supset\; \Diamond(s > 0)$$

by 4 and 3.

This covers case *b*.

We have

$$6. \quad \models \; [\Box\, at\,\ell_2 \wedge \sim at\,m_{2..5}] \;\supset\; (s = 1)$$

by Q_1.

$$7. \quad \models \; [\Box\, at\,\ell_2 \;\wedge\; \sim at\,m_{2..5}] \;\supset\; \Diamond(s > 0)$$

by 6.

This covers case *c*.

By taking the disjunction of 3, 5, and 7 we obtain

$$8. \quad \models \; \Box\, at\,\ell_2 \;\supset\; \Diamond(s > 0).$$

Applying the SEM rule at ℓ_2 yields

$$9. \quad \models \; at\,\ell_2 \;\supset\; \Diamond\, at\,\ell_3,$$

which is the desired Lemma *B*. ▮

EXAMPLE: BINOMIAL COEFFICIENT

We will now establish the termination of the program BC_1 for the distributed evaluation of a binomial coefficient. Since we have already proved the partial correctness of this program, termination will guarantee total correctness.

The statement to be proved is:

Theorem: $\models \; \Diamond(at\,\ell_e \wedge at\,m_e)$

The initial condition associated with the proper computation of the program is

$$at\,\ell_0 \;\wedge\; at\,m_0 \;\wedge\; (y_1 = n) \;\wedge\; (y_2 = 0) \;\wedge\; (y_3 = 1) \;\wedge\; (y_4 = 1) \;\wedge\; (0 \le k \le n).$$

We will use in our proof the following invariants that were established above:

$$\models \Box(Q_0 \wedge Q_1 \wedge Q_2),$$

where

$$Q_0 \text{ is } at\,\ell_{2..4} + at\,m_{4..6} + y_4 = 1$$
$$Q_1 \text{ is } ((n-k) \leq y_1 \leq n) \wedge (0 \leq y_2 \leq k)$$
$$Q_2 \text{ is } at\,\ell_e \supset (y_1 = n-k).$$

We start by proving a sequence of lemmas:

*Lemma A*1: $\models [at\,\ell_1 \wedge (y_1 = u)] \supset \Diamond[at\,\ell_2 \wedge (y_1 = u)]$

This lemma ensures that we never get stuck at ℓ_1 which is a semaphore instruction.

*Proof of Lemma A*1:

The proof distinguishes three cases according to the current location of P_2. In all cases we assume that P_1 is waiting at ℓ_1.

case a: P_2 is at m_6. The next time it will be scheduled will increment y_4, making it positive.

case b: P_2 is in $\{m_4, m_5\}$. Eventually it will get to m_6 and increment y_4.

case c: P_2 is in $\{m_0, m_1, m_2, m_3, m_7, m_e\}$. By Q_0 and the fact that P_1 is at ℓ_1, y_4 is currently positive.

In all three cases we can show that the value of y_1 never changes.

Thus we have:

$$\textbf{1.} \quad \models [\Box\, at\,\ell_1 \wedge at\,m_6] \supset [\Box\, at\,\ell_1 \wedge at\,m_6 \wedge (y_4 \geq 0)]$$

by Q_0.

$$\textbf{2.} \quad \models [\Box\, at\,\ell_1 \wedge at\,m_6 \wedge (y_4 \geq 0)] \supset \Diamond(y_4 > 0)$$

by the ESC rule at m_6 where ϕ is $y_4 \geq 0$, χ is $at\,\ell_1$.

$$\textbf{3.} \quad \models [\Box\, at\,\ell_1 \wedge at\,m_6] \supset \Diamond(y_4 > 0)$$

by 2 and 1.

This covers case *a*.

We have

$$\textbf{4.} \quad \models [\Box\, at\,\ell_1 \wedge at\,m_{4,5}] \supset \Diamond\, at\,m_6$$

by the SP rule applied to the path $m_4 \rightarrow m_5 \rightarrow m_6$.

$$5. \quad \models \; [\Box\, at\,\ell_1 \;\wedge\; at\,m_{4,5}] \;\supset\; \Diamond(y_4 > 0)$$

by 4 and 3.

This covers case *b*.

We have

$$6. \quad \models \; [\Box\, at\,\ell_1 \;\wedge\; \sim at\,m_{4..6}] \;\supset\; (y_4 > 0)$$

by Q_0. Therefore

$$7. \quad \models \; [\Box\, at\,\ell_1 \;\wedge\; \sim at\,m_{4..6}] \;\supset\; \Diamond(y_4 > 0)$$

This covers case *c*.

By taking the disjunction of 3, 5 and 7 we obtain

$$8. \quad \models \; \Box\, at\,\ell_1 \;\supset\; \Diamond(y_4 > 0).$$

Applying the SEM rule at ℓ_1 where ϕ is $y_1 = u$ we obtain

$$9. \quad \models \; [at\,\ell_1 \;\wedge\; (y_1 = u)] \;\supset\; \Diamond[at\,\ell_2 \;\wedge\; (y_1 = u)]. \quad \blacksquare$$

Lemma A2: $\models \; \{[at\,\ell_{1..5} \wedge (y_1 = u + 1)] \vee [at\,\ell_6 \wedge (y_1 = u)]\} \;\supset\; \Diamond[at\,\ell_0 \wedge (y_1 = u)]$

This lemma ensures that being anywhere in ℓ_1 to ℓ_5 we return to ℓ_0 with the value of y_1 smaller by 1 than the original and being at ℓ_6 we return to ℓ_0 with the value of y_1 unchanged.

Proof of Lemma A2:

After being ensured by Lemma *A*1 of not being blocked at ℓ_1 all that remains is to trace the value of y_1. Indeed:

$$1. \quad \models \; [at\,\ell_1 \;\wedge\; (y_1 = u + 1)] \;\supset\; \Diamond[at\,\ell_2 \wedge (y_1 = u + 1)]$$

by Lemma *A*1.

$$2. \quad \models \; \{[at\,\ell_{2..5} \wedge (y_1 = u + 1] \vee [at\,\ell_6 \wedge (y_1 = u)]\} \;\supset\; \Diamond[at\,\ell_0 \wedge (y_1 = u)]$$

by applying the SP rule to the path $\ell_2 \rightarrow \ell_3 \rightarrow \ell_5 \rightarrow \ell_6 \rightarrow \ell_0$ where $\phi_2 = \phi_3 = \phi_4 = \phi_5$ is $y_1 = (u + 1)$, ϕ_6 is $y_1 = u$, and ψ is $at\,\ell_0 \wedge (y_1 = u)$.

$$3. \quad \models \; [at\,\ell_1 \;\wedge\; (y_1 = u + 1)] \;\supset\; \Diamond[at\,\ell_0 \wedge (y_1 = u)]$$

by 1 and 2.

$$4. \quad \models \; \{[at\,\ell_{1..5} \wedge (y_1 = u + 1)] \vee [at\,\ell_6 \wedge (y_1 = u)]\} \;\supset\; \Diamond[at\,\ell_0 \wedge (y_1 = u)]$$

by 2 and 3.

This establishes Lemma $A2$. ▮

Lemma A3: $\models \; [at\,\ell_0 \wedge (y_1 \geq n-k)] \supset \Diamond[at\,\ell_e \wedge (y_1 = n-k)]$.

This lemma establishes the termination of P_1 if started at ℓ_0 with $y_1 \geq n-k$.

Proof of Lemma A3:

Define the auxiliary assertion:

$$E_1(u): \; [at\,\ell_0 \wedge (y_1 = u)] \supset \Diamond[at\,\ell_e \wedge (y_1 = n-k)].$$

We will establish the lemma by showing that

$$\models \; (u \geq n-k) \supset E_1(u).$$

This will be established by induction on $u \geq n-k$. We will have to show first

$$(a) \quad \models \; E_1(n-k)$$

and then

$$(b) \quad \models \; [(u \geq n-k) \wedge E_1(u)] \supset E_1(u+1).$$

(a) To prove part a we observe that $E_1(n-k)$ just says that if we are at ℓ_0 with $y_1 = n-k$ we will eventually get to ℓ_e with $y_1 = n-k$. This is obvious since when $y_1 = n-k$, P_1 proceeds directly from ℓ_0 to ℓ_e. Indeed:

$$\textbf{1.} \quad \models \; [at\,\ell_0 \wedge (y_1 = n-k)] \supset \Diamond[at\,\ell_e \wedge (y_1 = n-k)]$$

by the ESC rule applied at ℓ_0 where ϕ is $y_1 = n-k$ considering just the exit $\ell_0 \rightarrow \ell_e$ whose enabling condition c is $y_1 = n-k$. In other words,

$$\textbf{1}'. \quad \models \; E_1(n-k)$$

(b) To prove part b we assume that $u \geq n-k$ and $E_1(u)$ is true and consider an execution that starts at ℓ_0 with $y_1 = u+1$. Since $u+1 > n-k$ we will proceed to ℓ_1 with $y_1 = u+1$. By Lemma $A2$ we will return to ℓ_0 with $y_1 = u$. Now by the assumption of $E_1(u)$ we will eventually get to ℓ_e with $y_1 = n-k$.

For the formal proof, we assume:

$$\textbf{2.} \quad \models \; u \geq n-k$$

and

$$\textbf{3.} \quad \models \; E_1(u),$$

i.e.,

$$\textbf{3}'. \quad \models \; [at\,\ell_0 \wedge (y_1 = u)] \supset \Diamond[at\,\ell_e \wedge (y_1 = n-k)].$$

Then

4. $\models$ $[at\,\ell_0 \wedge (y_1 = u+1)] \supset [at\,\ell_0 \wedge (y_1 = u+1) \wedge (y_1 > n-k)]$

by 2.

5. $\models$ $[at\,\ell_0 \wedge (y_1 = u+1) \wedge (y_1 > n-k)] \supset \Diamond[at\,\ell_1 \wedge (y_1 = u+1)]$

by the ESC rule at ℓ_0 *using only* the $\ell_0 \rightarrow \ell_1$ exit where ϕ is $y_1 > n-k$.

6. $\models$ $[at\,\ell_0 \wedge (y_1 = u+1)] \supset \Diamond[at\,\ell_1 \wedge (y_1 = u+1)]$

by 4 and 5.

7. $\models$ $[at\,\ell_0 \wedge (y_1 = u+1)] \supset \Diamond[at\,\ell_0 \wedge (y_1 = u)]$

by 6 and Lemma $A2$.

8. $\models$ $[at\,\ell_0 \wedge (y_1 = u+1)] \supset \Diamond[at\,\ell_e \wedge (y_1 = n-k)]$

by 7 and 3′; *i.e.*, by the definition of E_1,

8′. $\models$ $E_1(u+1)$.

Applying the deduction theorem to 2, 3, and 8′, we obtain

9. $\models$ $(u \geq n-k) \supset [E_1(u) \supset E_1(u+1)]$.

Now we may combine parts *a* and *b* (*i.e.*, 1′ and 9) to deduce the lemma using the induction principle. ∎

Lemma A4: $\models$ $\Diamond[at\,\ell_e \wedge (y_1 = n-k)]$

This states that no matter where we are in a properly initialized execution of the program, we will eventually wind up at ℓ_e with $y_1 = n-k$.

***Proof of Lemma** A4*:

There are three cases to be considered according to the current location of P_1.

case a: P_1 is already at ℓ_e. Then we have by Q_2 that $y_1 = n-k$.

case b: P_1 is at ℓ_0. Then we are assured by Q_1 that $y_1 \geq n-k$; hence, by Lemma $A3$, we will wind up at ℓ_e with $y_1 = (n-k)$.

case c: P_1 is anywhere else, that is in $\{\ell_1, \ldots, \ell_6\}$. Then we will eventually get to ℓ_0 by Lemma $A2$, which is already covered by case *b*.

We proceed with the formal proof. We have

1. $\models$ $at\,\ell_e \supset [at\,\ell_e \wedge (y_1 = n-k)]$

by Q_2.

This corresponds to case *a*.

We have

$$2. \quad \models \; at\,\ell_0 \;\supset\; [at\,\ell_0 \wedge (y_1 \geq n-k)]$$

by Q_1.

$$3. \quad \models \; at\,\ell_0 \;\supset\; \Diamond[at\,\ell_e \wedge (y_1 = n-k)]$$

by Lemma $A3$.

This covers case *b*.

We have

$$4. \quad \models \; at\,\ell_{1..6} \;\supset\; \Diamond\, at\,\ell_0$$

by Lemma $A2$.

$$5. \quad \models \; at\,\ell_{1..6} \;\supset\; \Diamond[at\,\ell_e \wedge (y_1 = n-k)]$$

by 4 and 3.

This covers case *c*.

Taking the disjunction of 1, 3 and 5 we obtain

$$6. \quad \models \; \Diamond[at\,\ell_e \wedge (y_1 = n-k)]$$

which establishes the lemma. ∎

We now turn to the termination of P_2.

Lemma B0: $\models \; [at\,m_2 \wedge (y_2 = u)] \;\supset\; \Diamond[at\,m_3 \wedge (y_2 = u)]$

This lemma states that we can never get blocked at m_2.

Proof of Lemma B0:

By Lemma $A4$ we are guaranteed that P_1 will eventually get to ℓ_e with $y_1 = n-k$. In the worst case, by the time P_1 gets to ℓ_e, P_2 is still waiting at m_2. But then by Q_1, $y_2 \leq k$ and $y_1 = n-k$ so that $y_1 + y_2 \leq n$ which enables the exit condition and leaves it enabled until P_2 moves. This proof should not be considered as saying that P_2 will indeed wait at m_2 until P_1 terminates, but this approach provides the easiest proof.

Proceeding with more formal proof we have

$$1. \quad \models \; [at\,m_2 \wedge (y_2 = u)] \;\supset\; \{\Box[at\,m_2 \wedge (y_2 = u)] \vee \Diamond[at\,m_3 \wedge (y_2 = u)]\}$$

by the ALT rule at m_2 where ϕ is $y_2 = u$.

$$\mathbf{2.} \quad \models \ \Box[at\,m_2 \wedge (y_2 = u)] \ \supset \ \Diamond[at\,m_2 \wedge (y_2 = u) \wedge at\,\ell_e \wedge (y_1 = n - k)]$$

by Lemma $A4$.

$$\mathbf{3.} \quad \models \ [at\,m_2 \ \wedge \ (y_2 = u) \ \wedge \ at\,\ell_e \ \wedge \ (y_1 = n - k)] \\ \supset \ [at\,m_2 \ \wedge \ (y_2 = u) \ \wedge \ at\,\ell_e \ \wedge \ (y_1 + y_2 \leq n)]$$

using $y_2 \leq k$ given by Q_1.

$$\mathbf{4.} \quad \models \ [at\,m_2 \ \wedge \ (y_2 = u) \ \wedge \ at\,\ell_e \ \wedge \ (y_1 + y_2 \leq n)] \ \supset \ \Diamond[at\,m_3 \ \wedge \ (y_2 = u)]$$

by ESC at m_2 considering only the exit $m_2 \to m_3$ where ϕ is $(y_2 = u) \wedge at\,\ell_e \wedge (y_1 + y_2 \leq n)$.

$$\mathbf{5.} \quad \models \ \Box[at\,m_2 \wedge (y_2 = u)] \ \supset \ \Diamond[at\,m_3 \wedge (y_2 = u)]$$

by 2, 3, and 4.

$$\mathbf{6.} \quad \models \ [at\,m_2 \ \wedge \ (y_2 = u)] \ \supset \ \Diamond[at\,m_3 \wedge (y_2 = u)]$$

by 1 and 5. ▮

*Lemma B*1: $\models \ [at\,m_3 \ \wedge \ (y_2 = u)] \ \supset \ \Diamond[at\,m_4 \wedge (y_2 = u)]$

This lemma states that P_2 does not get blocked at m_3 but eventually proceeds to m_4 with an unchanged value of y_2.

It is analogous to Lemma $A1$ and has a very similar proof. In that proof we distinguish three cases according to the location of P_1. They are: P_1 at ℓ_4, P_2 in $\{\ell_2, \ell_3\}$, and P_2 elsewhere. Their analysis is identical to that of Lemma $A1$.

*Lemma B*2: $\models \ \{[at\,m_1 \ \wedge \ (y_2 = u)] \ \vee \ [at\,m_{2..7} \wedge (y_2 = u + 1)]\} \ \supset \ \Diamond[at\,m_0 \wedge (y_2 = u + 1)]$

This lemma states that if we are anywhere in m_1 to m_7 we will eventually return to m_0 with y_2 properly adjusted.

*proof of Lemma B*2:

$$\mathbf{1.} \quad \models \ [at\,m_{4..7} \ \wedge \ (y_2 = u + 1)] \ \supset \ \Diamond[at\,m_0 \wedge (y_2 = u + 1)]$$

by the SP rule applied to the path $m_4 \to m_5 \to m_6 \to m_7 \to m_0$ where $\phi_4 = \phi_5 = \phi_6 = \phi_7$ is $y_2 = u + 1$ and ψ is $at\,m_0 \wedge (y_2 = u + 1)$.

$$\mathbf{2.} \quad \models \ [at\,m_3 \ \wedge \ (y_2 = u + 1)] \ \supset \ \Diamond[at\,m_0 \wedge (y_2 = u + 1)]$$

by Lemma $B1$ and 1.

$$\mathbf{3.} \quad \models \ [at\,m_2 \ \wedge \ (y_2 = u + 1)] \ \supset \ \Diamond[at\,m_0 \wedge (y_2 = u + 1)]$$

by Lemma $B0$ and 2.

$$4. \quad \models \ [at\, m_1 \ \wedge \ (y_2 = u)] \ \supset \ \Diamond[at\, m_2 \wedge (y_2 = u+1)]$$

by the ESC rule at m_1 where ϕ is $y_2 = u$ and ψ is $at\, m_2 \wedge (y_2 = u+1)$.

$$5. \quad \models \ [at\, m_1 \ \wedge \ (y_2 = u)] \ \supset \ \Diamond[at\, m_0 \wedge (y_2 = u+1)]$$

by 4 and 3.

By taking the disjunction of 1, 2, 3 and 5 we obtain:

$$6. \quad \models \ \{[at\, m_1 \wedge (y_2 = u)] \ \vee \ [at\, m_{2..7} \wedge (y_2 = u+1)]\} \ \supset \ \Diamond[at\, m_0 \wedge (y_2 = u+1)].$$

Lemma B3: $\models \ [at\, m_0 \ \wedge \ (y_2 \leq k)] \ \supset \ \Diamond[at\, m_e \wedge (y = k)]$

This lemma establishes the termination of P_2 if started at m_0 with $y_2 \leq k$.

Proof of Lemma B3:

Similarly to the proof of Lemma $A3$ we define the auxiliary assertion

$$E_2(u): \ [at\, m_0 \ \wedge \ (y_2 = u)] \ \supset \ \Diamond[at\, m_e \wedge (y_2 = k)].$$

The lemma is established by showing that

$$\models \ (u \leq k) \ \supset \ E_2(u).$$

Analogously to A3 this is proven by descending induction on $u \leq k$. We show the two clauses:

$$(a) \quad \models \ E_2(k)$$

and

$$(b) \quad \models \ [(u < k) \ \wedge \ E_2(u+1)] \ \supset \ E_2(u).$$

Part a is proved by observing the direct path from m_0 to m_e in the case that $y_2 = k$. Part b is proved by tracing the execution from m_0 with $y_2 = u < k$ to m_1 with $y_2 = v + 1$ and use the induction hypothesis to finally guarantee $at\, m_e \wedge (y_2 = k)$.

The details of the formal proof are very similar to those of $A3$. ∎

Lemma B4: $\models \ \Diamond \ at\, m_e$

This statement says that regardless of where we are in a properly initialized execution of the program, we eventually wind up at m_e.

Proof of Lemma B4:

Similarly to the proof of Lemma $A4$ there are three cases to be considered:

case a: P_2 already at m_e.

case b: P_2 currently at m_0. Then we have by Q_1 that $y_2 \leq k$ and hence by Lemma $B3$ we will eventually reach m_e.

case c: P_2 is elsewhere. Then we will eventually get to m_0 by Lemma $B2$.

The formal details are similar to those of Lemma $A4$. ∎

Proof of Theorem:

To conclude the proof of the theorem we observe that:

$$1. \quad \models \ \ell_e \supset \Box\, at\, \ell_e$$

by the ALT rule since ℓ_e has no exits.

$$2. \quad \models \ \Diamond \Box\, at\, \ell_e$$

by Lemma $A4$ and 1. Similarly,

$$3. \quad \models \ \Diamond \Box\, at\, m_e$$

using Lemma $B4$ and the ALT rule at m_e.

A temporal consequence of 2 and 3 is

$$\models \ \Diamond[at\, \ell_e \wedge at\, m_e]. \quad ∎$$

Acknowledgement

We thankfully acknowledge the help extended to us by Yoni Malachi, Pierre Wolper, Frank Yellin, Joe Weening, and Rivi Zarhi in reading the earlier drafts of the manuscript. Special thanks are due to Evelyn Eldridge-Diaz for TEXing the manuscript.

REFERENCES

[BMP] Ben-Ari, M., Z. Manna and A. Pnueli, "The temporal logic of branching time," Proceedings of the Eighth ACM Symposium on Principles of Programming Languages, Williamsburg, VA, Jan. 1981, pp. 169-176.

[BUR] Burstall, R.M., "Program proving as hand simulation with a little induction," Proc. IFIP Congress, Amsterdam, The Netherlands (1974), North Holland, pp. 308-312.

[CLA] Clarke, E.M., "Synthesis of resource invariants for concurrent programs," ACM Trans. on Programming Languages and Systems, Vol. 2, No. 3 (July 1980), pp. 338-358.

[DIJ] Dijkstra, E.W., "Cooperating sequential processes", in *Programming Languages and Systems* (F. Genvys ed.), Academic Press, New York, NY, 1968, pp. 43 112.

[FRA] Francez, N., "The analysis of cyclic programs," Ph.D. Thesis, Applied Mathematics Dept., *The Weizmann Institute of Science, Rehovot, Israel, July 1976.*

[KEL] Keller, R.M., "Formal verification of parallel programs," CACM, Vol.19, No. 7 (July 1976), pp. 371-384.

[LAM] Lamport, L., "Proving the correctness of multiprocess programs," IEEE Transactions on Software Engineering, Vol. SE-3, No. 7 (March 1977), pp. 125-143.

[MAN1] Manna, Z., "Logics of programs," Proc. IFIP Congress, Tokyo and Melbourne (October 1980), North Holland, pp. 41-51.

[MAN2] Manna, Z., "Verification of sequential programs: Temporal axiomatization" in *Theoretical Foundations of Programming Methodology* (F.L. Bauer, ed.), NATO Scientific Series, D. Riedel Pub. Co., Dordrecht, Holland, 1981. Also, Computer Science Report, Stanford University, Stanford, CA (October 1981).

[MP1] Manna, Z. and A. Pnueli, "The modal logic of programs," Proc. 6th International Colloquium on Automata, Languages and Programming, Graz, Austria (July 1979). Lecture Notes in Computer Science, Vol. 71, Springer Verlag, pp. 385-409.

[MP2] Manna, Z. and A. Pnueli, "Verification of concurrent programs: The temporal framework," in *The Correctness Problem in Computer Science* (R.S. Boyer and J S. Moore, eds.), International Lecture Series in Computer Science, Academic Press, London, 1981. Also, Computer Science Report, Stanford University, Stanford, CA (June 1981).

[MW] Manna, Z. and R. Waldinger, "Is 'sometime' sometimes better than 'Always'?: Intermittent assertions in proving program correctness," CACM, Vol. 21, No. 2, pp. 159-172 (February 1978), pp. 159-172.

[OG] Owicki, S. and D. Gries, "An axiomatic proof technique for parallel programs," Acta Informatica, Vol. 6 (1976), pp. 319-340.

[OL] Owicki, S. and L. Lamport, "Proving liveness properties of concurrent programs," unpublished report (october 1980).

[PNU1] Pnueli, A., "The temporal logic of programs," Proc. 18th FOCS, Providence, RI (November 1977), pp. 46-57.

[PNU2] Pnueli, A., "The temporal semantics of concurrent programs," Proc. Symposium on Semantics of Concurrent Computations, Evian, France (July 1979), Lecture Notes in Computer Science,Vol. 70, Springer Verlag, pp. 1-20.

DAT
ILM